

# Learning to Update Knowledge Graphs by Reading News

Jizhi Tang, Yansong Feng, Dongyan Zhao

Institute of Computer Science and Technology, Peking University  
The MOE Key Laboratory of Computational Linguistics, Peking University  
{tangjizhi, fengyansong, zhaody}@pku.edu.cn

## Abstract

News streams contain rich up-to-date information which can be used to update knowledge graphs (KGs). Most current text-based KG updating methods rely on elaborately designed information extraction systems and carefully crafted rules, which are often domain-specific and hard to maintain or generalize. However, such methods may not pay enough attention to the implicit information that lies underneath texts, thus often suffer from coverage issues. In this paper, we propose a novel graph based neural network method, GUpdater, to tackle these problems<sup>1</sup>. GUpdater is built upon graph neural networks (GNNs) with a text-based attention mechanism to guide the updating message passing through the KG structures. Experiments on a real-world KG updating dataset show that our model can effectively broadcast the news information to the KG structures and perform necessary link-adding or link-deleting operations to ensure the KG up-to-date according to news snippets.

## 1 Introduction

Knowledge graphs have been widely used in different areas, where keeping the knowledge triples up-to-date is a crucial step to guarantee the KG quality.

Existing works attempt to synchronize KGs with encyclopedia sources (Morsey et al., 2012; Liang et al., 2017), which basically leverage structured data, while directly updating KGs using plain texts, such as news snippets, is not an easy task and still remains untouched. For example, given a KG recording the rosters of all NBA teams, we need to update this KG according to a news article which describes a trade between different teams. Given the following news snippet that reports a trade between Minnesota Timberwolves

and Philadelphia 76ers, one has to add and delete many links in the KG, as illustrated in Figure 1.

The Minnesota Timberwolves has acquired forward Robert Covington, forward Dario Šarić, guard Jerryd Bayless and a 2022 second-round draft pick from the Philadelphia 76ers in exchange for forward Jimmy Butler and center Justin Patton.

Current approaches either rely on manual updating, or often solve it in a two-step manner: first, using an off-the-shelf information extraction (IE) tool to extract triples from the text, and then modifying the KG according to certain rules predefined by domain experts. However, we should notice that besides what we can explicitly extract from the text, the implicit information, i.e., information that is not mentioned in the text but can be inferred from the text, should also be taken into consideration. This requires far more complicated updating rules. In the above example, if Robert Covington (RC) is traded from Philadelphia 76ers to Minnesota Timberwolves, his teammates should all be changed accordingly, which is not mentioned in the news at all. This is what we referred as implicit information in this scenario.

We should point out that both explicit and implicit information are just different aspects of the same event, which is different from another stream of research that focuses on the evolution of graphs (Pareja et al., 2019; Jin et al., 2019). For example, changing the head coach may trigger a possible trade for his/her favorite players in the future, but these are actually two events, thus beyond the scope of our work.

Intuitively, the implicit information behind the news is related to the KG structures. Many recent works focus on embedding a KG into a continuous vector space, which can be used to fill in miss-

<sup>1</sup>Code and data are available at: <https://github.com/esdse/GUpdater>

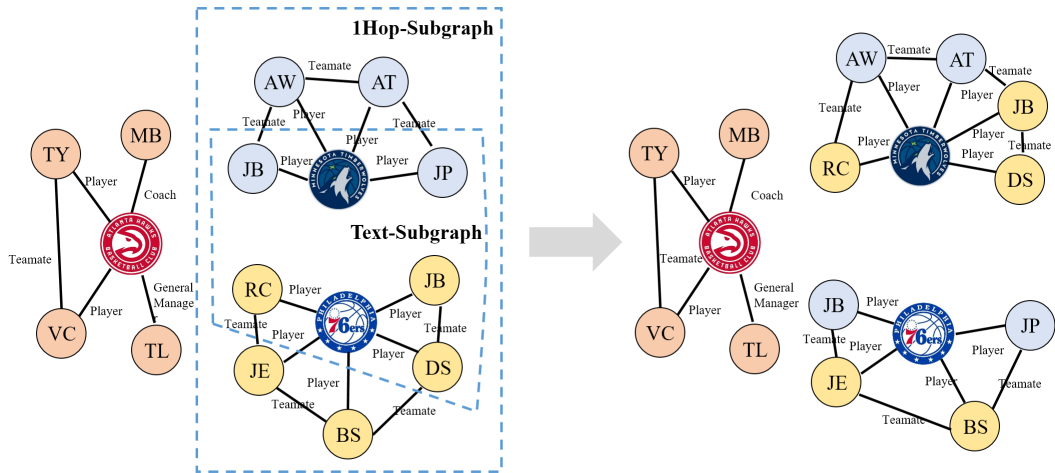


Figure 1: An example illustrating a trade in the KG of NBA teams. Note that all changes described by the news happen in the text-subgraph, the implicit changes, e.g., the teammate changes of player Robert Covington (RC), locate outside the text-subgraph but inside the 1hop-subgraph. The outside of the 1hop-subgraph, e.g., the Atlanta Hawks part, will not be affected by this trade.

ing links in KGs (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Yang et al., 2014; Trouillon et al., 2016; Schlichtkrull et al., 2018). However, most of them learn from static KGs, thus unable to help with our task, since they can not dynamically add new links or delete obsolete links according to extra text information.

In this paper, we propose a novel neural model, GUpdater, to tackle this problem, which features a graph based encoder to learn latent KG representations with the guidance from the news text, and a decoder to score candidate triples with reconstructing the KG as the objective. Our encoder is built upon the combination of recently proposed R-GCN (Schlichtkrull et al., 2018) and GAT (Velickovic et al., 2017) with two new key factors designed specifically for our task. The main idea is to control the updating message from the text passing through the KG. First, we use the given text to generate all the graph attention weights in order to selectively control message passing. Second, we link all entities mentioned in the news text together as shortcuts in order to let the message pass to each other even if they are topologically far from each other in the KG. For the decoder, we simply use DistMult (Yang et al., 2014) to score related triples to be updated. To evaluate our method, we construct a new real-world dataset, NBAtransactions, for this task. Experimental results show that our model can effectively use the news text to dynamically add new links and remove obsolete links accordingly.

Our contributions are in two-fold:

1. we propose a new text-based KG updating task and release a new real-world dataset to the community for further research.
2. we design a novel neural method, GUpdater, to read text snippets, which features an attention mechanism to selectively control the message passing over KG structures. This novel architecture enables us to perform both link-adding and link-deleting to ensure the KG up-to-date.

## 2 Task Formulation

We first formally define the task. Given a knowledge graph  $\mathcal{G} = (\mathbf{E}, \mathbf{R}, \mathbf{T})$ , where  $\mathbf{E}$ ,  $\mathbf{R}$  and  $\mathbf{T}$  are the entity set, relation set and KG triple set, respectively, and a news text snippet  $S = \{w_1, w_2, \dots, w_{|S|}\}$ , for which entity linking has been performed to build the mentioned entity set  $\mathbf{L} \subset \mathbf{E}$ , the text-based knowledge graph updating task is to read the news snippet  $S$  and update  $\mathbf{T}$  accordingly to get the final triple set  $\mathbf{T}'$  and the updated graph  $\mathcal{G}' = (\mathbf{E}, \mathbf{R}, \mathbf{T}')$ . In this paper, we focus on the scenarios where the entity set remains unchanged.

As a single event can only affect the KG in a limited range, for each event that is described by a news snippet, we consider two kinds of subgraphs that are defined by their entity sets, as illustrated in Figure 1:

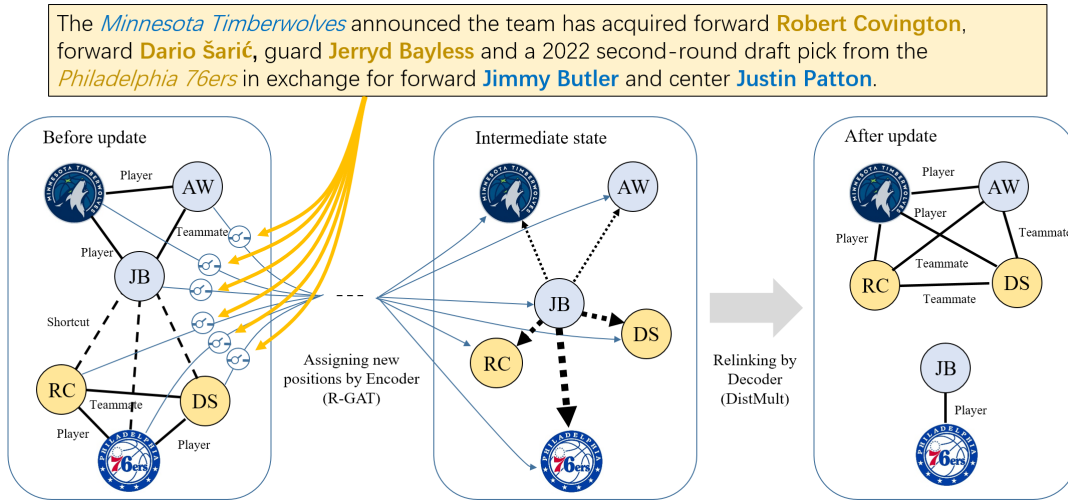


Figure 2: Overview of our GUpdater. By adding shortcuts in the R-GAT encoder, we enable Jimmy Butler (JB) to directly receive information from Philadelphia 76ers. Then with the guidance of text-based attention, Jimmy Butler selectively gathers neighbors’ messages and moves closer to Philadelphia 76ers in the latent space. Dashed lines with arrow in the intermediate state indicate the moving directions, and the width of the dash line illustrates how possible Jimmy Butler will move to that direction. Finally, new links are generated by the DistMult decoder.

**Text-Subgraph** Subgraphs with entity set  $\mathbf{E}_{text} = \mathbf{L}$ , i.e., all the entities are mentioned in news snippet  $S$ . Explicit information in the texts can lead to changes in these graphs only.

**1Hop-Subgraph** If we expand text-subgraphs by one hop, we get 1hop-subgraphs. For each entity in 1hop-subgraphs, either itself or one of its neighbors is mentioned in the news. The one-hop entity set is formally defined as  $\mathbf{E}_{1hop} = \{e | e \in \mathbf{L} \vee ((e, r, t) \in \mathbf{T} \vee (t, r, e) \in \mathbf{T}) \wedge t \in \mathbf{L}\}$ .

We argue that updating in 1hop-subgraphs only is a good balance of effectiveness and efficiency. On the one hand, implicit changes always occur around explicit changes usually within a short distance. In fact, most of the implicit changes can be captured in the one-hop range. Here we give an extreme example: given the news snippet “Barack Obama was elected president.”, the explicit change is to add (Barack Obama, Position, President) to the KG, while one of the implicit changes is to add (Michelle Obama, Position, First Lady) to the KG. Note that in this case, both “Michelle Obama” and “First Lady” are not mentioned in the news, but, because they are neighbors of “Barack Obama” and “President”, respectively, this implicit change can be found in the 1hop-subgraph. On the other hand, updating in small subgraphs can avoid a huge amount of meaningless computation, and meanwhile can reduce mis-prediction. For these two reasons, we se-

lect the one-hop range, i.e., the 1hop-subgraph as the main testbed for this task. Also note that this one-hop setting can be easily extended to two-hop or larger scopes if necessary.

### 3 Our Method

The overview of our model is illustrated in Figure 2. As mentioned before, our GUpdater follows an encoder-decoder framework with the objective to reconstruct the modified KGs.

#### 3.1 Relational Graph Attention Layer

In order to better capture the KG structures, we propose a new graph encoding layer, the relational graph attention layer (R-GAT), as the basic building block of GUpdater’s encoder. R-GAT can be regarded as a combination of R-GCN (Schlichtkrull et al., 2018) and GAT (Velickovic et al., 2017), which benefits from both the ability of modeling relational data and the flexibility of attention mechanism. Recall that the layer-wise propagation rule of R-GCN can be written as follows:

$$\mathbf{H}^{l+1} = \sigma \left( \sum_{r \in \mathbf{R}} \hat{\mathbf{A}}_r^l \mathbf{H}^l \mathbf{W}_r^l \right), \quad (1)$$

where  $\hat{\mathbf{A}}_r^l$  is the normalized adjacent matrix for relation  $r$ ,  $\mathbf{W}_r^l$  is a layer-specific trainable weight matrix for relation  $r$ , and  $\sigma(\cdot)$  denotes an activa-

tion function, here we use ReLU.  $H^l$  is the latent entity representations in the  $l^{th}$  layer.

Upon R-GCN, we can easily introduce attention mechanisms by computing  $\hat{\mathbf{A}}_r$  using an attention function:

$$a_{ij}^{lr} = \begin{cases} \frac{\exp(\text{att}^{lr}(\mathbf{h}_i^l, \mathbf{h}_j^l))}{\sum_{k \in \mathcal{N}_i^r} \exp(\text{att}^{lr}(\mathbf{h}_i^l, \mathbf{h}_k^l))} & , j \in \mathcal{N}_i^r \\ 0 & , \text{otherwise} \end{cases} \quad (2)$$

where  $a_{ij}^{lr}$  is the  $i^{th}$  row and  $j^{th}$  column element of  $\hat{\mathbf{A}}_r^l$ , and  $\mathcal{N}_i^r$  denotes the set of neighbor indices of node  $i$  under relation  $r$ .  $\text{att}^{lr}(\cdot, \cdot)$  is the attention function.  $\mathbf{h}_i^l$  and  $\mathbf{h}_j^l$  are the  $i^{th}$  and the  $j^{th}$  entity representations of layer  $l$ , respectively.

A known issue for R-GCN (Eq.(1)) is that the number of model parameters grows rapidly with the increasing number of relation types. We thus use *basis-decomposition* (Schlichtkrull et al., 2018) for regularization. In a standard R-GCN layer, if there are  $R$  relations involved, there will be  $R$  weight matrices for each layer. Basis-decomposition regularizes these matrices by defining each weight matrix as a linear combination of  $B$  ( $B < R$ ) basis matrices, which significantly decreases the parameter number.

### 3.2 Text-based Attention

The core idea of GNNs is to gather neighbors' information. In our case, we propose a text-based attention mechanism to utilize the news snippet to guide the message passing along the KG structure within the R-GAT layer.

We first use bi-GRU (Cho et al., 2014) to encode the given news text  $S$  into a sequence of representations  $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{|S|}\}$ , then we leverage the sequence attention mechanism (Luong et al., 2015) to compute the context vector:

$$\mathbf{c}^{lr} = \sum_{t=1}^{|S|} b_t^{lr} \mathbf{u}_t \quad (3)$$

where  $b_t^{lr}$  is the text attention weight, and is computed as follow:

$$b_t^{lr} = \frac{\exp(\mathbf{u}_t^T \mathbf{g}_{text}^{lr})}{\sum_{k=1}^{|S|} \exp(\mathbf{u}_k^T \mathbf{g}_{text}^{lr})} \quad (4)$$

where  $\mathbf{g}_{text}^{lr}$  is a trainable guidance vector to guide the extraction of relation-dependent context.

Recall that in GAT, the way to compute the attention weights is similar to the formula below:

$$\text{att}^{lr}(\mathbf{h}_i^l, \mathbf{h}_j^l) = \mathbf{g}_{graph}^{lr}{}^T [\mathbf{h}_i^l || \mathbf{h}_j^l] \quad (5)$$

where  $||$  denotes concatenation operation, and  $\mathbf{g}_{graph}^{lr}$  is a relation-specific trainable vector that serves as a graph guidance vector to decide which edge to pay attention to.

Here, we generate the final guidance vector  $\mathbf{g}_{fin}^{lr}$  that combines textual information to the graph guidance vector by simply using linear interpolation, and we replace  $\mathbf{g}_{graph}^{lr}$  in Eq.(5) by  $\mathbf{g}_{fin}^{lr}$  to get our final attention function:

$$\mathbf{g}_{fin}^{lr} = \alpha^{lr} \mathbf{g}_{graph}^{lr} + (1 - \alpha^{lr}) \mathbf{U}^{lr} \mathbf{c}^{lr}, \quad (6)$$

$$\text{att}^{lr}(\mathbf{h}_i^l, \mathbf{h}_j^l) = \mathbf{g}_{fin}^{lr}{}^T [\mathbf{h}_i^l || \mathbf{h}_j^l] \quad (7)$$

where  $\mathbf{U}^{lr}$  is a trainable transformation matrix, and  $\alpha^{lr} \in [0, 1]$  can either be trainable or fixed. If we set  $\alpha^{lr} = 1$ , then Eq.(7) degenerates to Eq.(5) and our encoder degenerates to R-GAT. It makes it easy to pre-train the model to get good embeddings for all entities and relations when news snippets are not provided.

### 3.3 Shortcuts

**Shortcuts** In practice, entities in  $\mathbf{L}$  can be far from each other in the KG, even unreachable sometimes, while a single- or two-layer R-GAT can only gather information from a near neighborhood. In order to encourage more direct interactions through the graph structure, before running the model, we simply link all entities in  $\mathbf{L}$  to each other and assign these links with a new relation label, *SHORTCUT*.

**Shortcuts with labels** Actually, relations between entities in  $\mathbf{L}$  can be different. For example, if a player is traded from one team to another, the relations between this player and that two teams must be opposite. Thus a unified shortcut label may make it difficult to correctly pass opposite messages, even with the help of the attention mechanism. So we further extend the shortcuts' label to 3 types: *ADD*, *DEL* and *OTHER*. *ADD* (*DEL*) denotes that these triples are explicitly mentioned in the text to be added to (deleted from) the KG. The rest are labeled with *OTHER*. Off-the-shelf IE tools can be used to generate these labeled shortcuts.

Type	Data type distribution		Edge number distribution (average number of edges in one instance)					
	Count	Percentage	Added	Deleted	Unchanged	T-added	T-deleted	T-unchanged
Trade	1245	30.4%	37.48	37.19	330.51	2.08	2.09	0.50
Head coach	77	1.9%	1.0	1.0	217.0	1.0	1.0	0.14
Free agency	1078	26.3%	19.19	0.0	179.89	1.0	0.0	0.0
Draft	137	3.3%	18.84	0.0	173.42	1.0	0.0	0.0
Released	1382	33.7%	0.0	18.65	170.26	0.0	1.0	0.0
Overseas	85	2.1%	0.0	17.62	150.05	0.0	1.0	0.0
Retirement	57	1.4%	0.0	16.51	132.07	0.0	1.0	0.0
D-league	39	1.0%	0.0	19.13	178.0	0.0	1.0	0.0
Overall	4100	100%	17.07	18.37	221.56	0.95	1.03	0.15

Table 1: The Statistics of NBAtransactions. Added, Deleted and Unchanged represent the average number of edges that should be added to, deleted from and remain unchanged in the KG, respectively. T-added represents the average edge addition in the text-subgraphs, similar for T-deleted and T-unchanged.

Here, we easily build a simple extraction model using our GUpdater encoder. We get the entity representations from GUpdater’s encoder (with unified shortcuts), then for each entity pair  $(e_i, e_j)$ , where  $e_i, e_j \in \mathbf{L}, i \neq j$ , we use an MLP classifier to get the probability of each label:

$$P(z) = \text{softmax}(\text{MLP}([\mathbf{h}_i || \mathbf{h}_j])) \quad (8)$$

where shortcut label  $z \in \{ADD, DEL, OTHER\}$ ,  $\mathbf{h}_i$  and  $\mathbf{h}_j$  are entity representations for entity  $e_i, e_j$ , respectively.

In our experiments, we train the shortcut-labeling module separately, as we find that the results are good enough while saving training time. One can surely perform the extraction step in a joint training fashion.

### 3.4 Decoder

We use DistMult (Yang et al., 2014), which is known to have good performance on standard KG completion tasks, followed by sigmoid function, as the decoder, i.e., for each possible triple  $(e_i, r_k, e_j)$  where  $e_i, e_j \in \mathbf{E}_{1hop}, r_k \in \mathbf{R}$ , the probability of this triple to appear in the final KG is computed as follow:

$$P(y) = \text{sigmoid}(\mathbf{h}_i^T(\mathbf{r}_k \circ \mathbf{h}_j)) \quad (9)$$

where  $\circ$  denotes element-wise multiplication.

Since we formulate KG updating as a binary classification task, we use the cross-entropy loss to train the model.

## 4 Experiments

### 4.1 Dataset

We construct a new dataset, NBAtransactions, to evaluate our method. NBAtransactions contains 4,100 transaction-news pairs in NBA from 2010

to 2019, 3,261 for training, 417 for validation and 422 for testing. We consider 8 different kinds of transactions, which can be divided into 3 categories according to the updating patterns: 1) *Adding edges only*: free agency and draft. 2) *Deleting edges only*: released, overseas, retirement and d-league. 3) *Both adding and deleting edges*: trade and head coach.

For each transaction, a news snippet  $S$  with mentioned entity set  $\mathbf{L}$  and two undirected KG fragments,  $\mathcal{G}_{1hop}$  and  $\mathcal{G}'_{1hop}$ , representing the corresponding 1hop-subgraphes before and after the transaction, respectively, are given. Averagely, one subgraph contains 27.86 entities, 4 types of relations, and 239.28 triples, and one news snippet contains 29.10 words and 2.66 KG entities. Each transaction averagely causes adding of 17.07 edges and deleting of 18.37 edges; among them, only 0.95 (for adding) and 1.03 edges (for deleting) are in the text-subgraph, i.e., only 5.6% of the edge changes are explicitly mentioned in the news texts. Detailed statistics are shown in Table 1.

Our dataset is collected from several NBA related websites. The KGs we used in the dataset are constructed using the roster of each NBA team in each season collected from Basketball-Reference<sup>2</sup>. For each NBA season, we build a large KG that records all teams, players, head coaches, general managers and their relations at the beginning of that season. So there are 9 large KGs in total, one KG for one NBA season. Wikipedia<sup>3</sup> records all NBA transactions from 1946 to 2019 in a structured form and provides URLs of news sources for most of the transactions after 2010. We crawled all the available

<sup>2</sup><https://www.basketball-reference.com>

<sup>3</sup>[https://en.wikipedia.org/wiki/Category:NBA\\_transactions](https://en.wikipedia.org/wiki/Category:NBA_transactions)

news and corresponding transactions. For each transaction,  $\mathcal{G}_{1hop}$  can be easily extracted from the corresponding large KG. However, we cannot get  $\mathcal{G}'_{1hop}$  directly, as our KGs only record the rosters at the beginning of each season. To generate the  $\mathcal{G}'_{1hop}$ , we manually create a large set of complicated conversion rules for each transaction type to modify  $\mathcal{G}_{1hop}$  and obtain the 1hop-subgraph after the transaction. We use string matching algorithms to perform entity linking, and meanwhile generate the mentioned entity set  $\mathbf{L}$ .

## 4.2 Setup

The dimensions of all embeddings (words, entities, and relations) are all set to 128, and the hidden dimension is 256. We use a single layer encoder, as we find that more layers do not bring any benefit. The basis number of basis-decomposition is 2. We replace the word embeddings of entity mentions in the text by the entity embeddings for better alignment of word embedding space and entity embedding space. The entity embeddings and relation embeddings are pre-trained using R-GAT, and the word embeddings are randomly initialized. We set dropout (Srivastava et al., 2014) rate to 0.5. The batch size in our experiments is 1. We use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 for training.

We consider updating in two different scopes: 1) Text-Subgraphs: changes in these subgraphs correspond to the explicit information mentioned in the texts. 2) 1Hop-Subgraphs: all explicit and implicit changes happen in these subgraphs, thus this setting is treated as the overall/real-world updating evaluation.

## 4.3 Metrics

As our model aims to reconstruct the KG using binary classification, i.e., deciding whether a possible triple should be in the modified KG, we use accuracy, precision, recall and F1 score as the evaluation metrics. Further, to evaluate the ability of link-adding, link-deleting and link-preserving, respectively, we collect all added edges, deleted edges, and unchanged edges and compute the prediction accuracies separately, which we denote as Added Acc, Deleted Acc and Unchanged Acc, respectively.

## 4.4 Baseline Models

Because most current text-based KG updating methods are in two steps: first extract information

from texts, then add and remove links in KGs, we select non-rule-based models that perform well on these two steps and also their combination as our baseline models.

**PCNN** (Zeng et al., 2015) is a strong baseline for relation extraction, which divides the sentence into three pieces and applies max-pooling in a piecewise manner after the convolution layer.

**IE-gold** is a simulation of an ideal IE model that can perfectly extract explicit information from given texts. This is an upper bound for the information extraction step.

**DistMult** (Yang et al., 2014) is a widely-used multiplication-based triple scoring function for KG completion, which computes the three-way inner-product of the triples. Its symmetric nature is suitable for NBA transactions as the KGs are undirected.

**R-GCN** (Schlichtkrull et al., 2018) is one of the strongest baselines for KG completion, which first encodes entities by gathering neighbors' information, then decodes them by DistMult. Note that compared to traditional KG completion tasks, here the target KGs are the modified KGs and usually different from the input KGs. Because of the lack of network structure, single DistMult is unable to train like this.

**IE-gold + R-GCN** is the combination of IE-gold and R-GCN. We first use IE-gold to generate a perfectly updated text-subgraph, then use R-GCN to predict other changes outside the text-subgraph.

## 4.5 Main Results

We summarize the main results in Table 2. As we can see, PCNN performs pretty well in extracting explicit information from the news (over 0.96 in Added Acc, Deleted Acc and Unchanged Acc in Text-Subgraphs). However, the explicit changes only take up 5.8% in the testing set, while the majority of changes are implicit. Therefore, even a perfect IE model, IE-gold, which never makes any wrong predictions, can only correctly predict about 6% of the changes in the 1hop-subgraphs.

The implicit changes are highly related to KG structures. Compared to DistMult, which just preserves the original structures and scores 0.0656 in Added Acc, 0.1220 in Deleted Acc in the 1Hop-Subgraphs, R-GCN can use the modified KGs for training and learns to predict changes without reading the news. R-GCN beats DistMult by

	Accuracy	Precision	Recall	F1	Added Acc	Deleted Acc	Unchanged Acc
<b>Text-Subgraphs</b>							
PCNN	0.9664	0.9573	0.9676	0.9624	0.9674	0.9638	0.9686
IE-gold	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DistMult	0.8372	0.1640	0.1670	0.1655	0.0679	0.0652	0.7093
R-GCN	0.8601	0.3249	0.4147	0.3644	0.4055	0.2077	0.4651
IE-gold + R-GCN	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GUpdater	0.9783	0.8871	0.8887	0.8879	0.8684	0.9185	0.9651
<b>1Hop-Subgraphs</b>							
PCNN	0.9799	0.9245	0.9270	0.9257	0.0586	0.0541	0.9995
IE-gold	0.9805	0.9248	0.9274	0.9261	0.0606	0.0561	1.0
DistMult	0.9644	0.8277	0.9225	0.8725	0.0656	0.1220	0.9944
R-GCN	0.9837	0.9243	0.9547	0.9393	0.4348	0.2005	0.9984
IE-gold + R-GCN	0.9847	0.9285	0.9577	0.9429	0.4681	0.2448	0.9989
GUpdater	0.9964	0.9819	0.9913	0.9866	0.8926	0.8988	0.9991

Table 2: Model performance on NBAtransactions. Here, PCNN performs multi-class classification while others are binary classification models.

0.37 in Added Acc and 0.08 in Deleted Acc in the 1Hop-Subgraphs. Such a huge improvement suggests that the blind prediction is not completely arbitrary, and R-GCN indeed learns certain updating patterns through the KG structures. Besides, R-GCN scores 0.9393 in F1 in the 1hop-subgraphs, outperforms IE-gold by 1%, once again underlines the importance of implicit changes.

IE-gold + R-GCN combines the advantages of IE-gold and R-GCN, and performs best among the baselines. However, in the 1hop-subgraphs, the Added Acc and Deleted Acc are only 0.4681 and 0.2448, which are still quite low. Although IE-gold + R-GCN can perfectly extract explicit information from the news, there is no mechanism for the IE model to pass the extracted information to the R-GCN module, thus this two-step method still performs poorly in finding out the implicit changes.

For overall performance, GUpdater significantly outperforms all baseline models, which coincides to our intuition. Particularly, in the 1hop-subgraphs, GUpdater beats IE-gold + R-GCN by 0.42 in Added Acc, 0.65 in Deleted Acc, indicating that GUpdater can well capture both explicit and implicit information in the news snippets.

#### 4.6 Ablation Analysis

We also perform ablation test to figure out the effect of each component of GUpdater. As shown in Table 3, compared to R-GAT, both GUpdater-shortcut (GUpdater without shortcut) and GUpdater-text (GUpdater without text-based attention) improve the overall performance with different levels, showing that both the text-based attention and the shortcuts are helpful for this task.

Generally, we can see that adding shortcuts is key to this task, as it brings a giant leap to nearly all indicators. Actually, updating KGs using news can be regarded as information injection from one semantic space to another, the text mentioned entities can be seen as junctions of these two spaces and serve as entrances for information injection. So, in the KG, the text mentioned entities are information sources, and send messages to the explicitly and implicitly related target entities. However, the targets may be very far and hard to reach, it is the shortcuts that make the successful information delivery possible.

However, if there are too many shortcuts, the messages are easy to get to wrong targets. The text-based attention mechanism can be regarded as gates that selectively open several shortcuts that are easy for message passing, and this helps the model get 2%-3% steady improvements in both Added Acc and Deleted Acc.

When adding explicit labels to shortcuts, we get much better performance than the basic GUpdater (7% improvement in Added Acc, 8% improvement in Deleted Acc in 1hop-subgraphs), which indicates that splitting the shortcuts into different channels for different kinds of messages is crucial for better information delivery, and that makes it easier to dig out implicit information. The result also indicates that our model is well extensible and can easily incorporate off-the-shelf IE tools.

#### 4.7 Visualization of Attention on Shortcuts

In order to explore how the information actually passes along shortcuts, we select 5 different types of trades, as shown in Table 4, and we get the corresponding attention weight on each shortcut in

	Accuracy	Precision	Recall	F1	Added Acc	Deleted Acc	Unchanged Acc
<b>Text-Subgraphs</b>							
GUdater	0.9783	0.8871	0.8887	0.8879	0.8684	0.9185	0.9651
-shortcut-text (R-GAT)	0.8623	0.3343	0.4273	0.3751	0.4225	0.2179	0.4535
-shortcut	0.8656	0.3551	0.4776	0.4074	0.2059	0.5112	0.3372
-text	0.9705	0.8550	0.8366	0.8457	0.8323	0.8982	0.8605
GUdater+label	0.9936	0.9779	0.9551	0.9664	0.9584	0.9603	0.9862
<b>1Hop-Subgraphs</b>							
GUdater	0.9964	0.9819	0.9913	0.9866	0.8926	0.8988	0.9991
-shortcut-text (R-GAT)	0.9833	0.9216	0.9547	0.9378	0.4290	0.2073	0.9990
-shortcut	0.9843	0.9267	0.9563	0.9413	0.4726	0.2044	0.9989
-text	0.9957	0.9793	0.9885	0.9839	0.8668	0.8619	0.9987
GUdater+label	0.9981	0.9889	0.9969	0.9929	0.9668	0.9792	0.9991

Table 3: Results of the ablation test, where -text indicates removing the text-based attention mechanism from GUdater.

GUdater. We find that most shortcuts are closed by the attention mechanism, i.e., their corresponding attention weights are very close to 0, and only a small portion of shortcuts are open for message passing.

For each trade type, the remaining shortcuts are organized in a similar pattern: there is a central team selected by the model that sends messages to all other entities and receives the message from another team. For symmetric trades, i.e., each team plays the same role in the trade (T2, T3 in Table 4), the selection of the central team seems to be arbitrary, while for asymmetric trades (T4, T5), the central teams are the teams that involved in more transactions.

It seems that GUdater learns a 2-team trade rule from that pattern: if two teams send messages mutually through shortcuts, they will exchange all their players in text-subgraphs. T1 and T2 are most simple 2-team trades, thus our model updates the graphs perfectly with this message passing pattern. T4 and T5 look difficult and complicated, but they are actually direct combinations of two simple 2-team trades, e.g., T4 can be decomposed into two trades: the Grizzlies trades JS to the Celtics for FM, and the Thunder trades RG to the Celtics. So for these trades, GUdater also performs quite well, as only two not-that-important triples are missing in T5. However, in T3, three teams exchange players in a rotation, and it cannot be decomposed into several 2-team trades, thus leads to a severe mis-prediction: (TS, player, Jazz), as placing a player in a wrong team may cause more teammate-errors in the 1hop-subgraph. The inability of perfectly performing rotational three-team trades may due to the lack of training instance, as in the past 10 years, such kind of trade

happened nearly once a year.

## 5 Related Work

**KG Representation Learning** aims to embed a KG into a continuous vector space, which preserves the KG structures. There are mainly two streams of researches. The first is addition-based models (also called translation-based models), which based on the principle that for every valid triple  $(h, r, t)$ , their embeddings holds:  $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ . **TransE**(Bordes et al., 2013) is the first such model. **TransH**(Wang et al., 2014) projects entity embeddings into relation-specific hyperplanes. **TransR**(Lin et al., 2015) generalize TransH by extending projection to linear transformation. **TransD**(Ji et al., 2015) simplifies TransR by decomposing the transformation matrix into the product of two vectors. Another is multiplication-based models, which comes from the idea of tensor decomposition. **RESCAL**(Nickel et al., 2011) is one of the earliest studies, which using a bilinear scoring function. **DistMult**(Yang et al., 2014) simplifies RESCAL by using a diagonal matrix. **Complex**(Trouillon et al., 2016) extend DistMult into the complex space to handle asymmetric relations. **Simple**(Kazemi and Poole, 2018) learns two embeddings for each entity dependently. A direct application is KG completion.

**Graph Neural Networks** allow passing information horizontally among nodes (Gilmer et al., 2017). Recently, many techniques has been incorporated to GNN model, and lots of attempt has been made to promote GNNs to more application scenarios. **GCN**(Kipf and Welling, 2016) leverages graph spectrums for semi-supervised node classification. **GAT**(Velickovic et al., 2017) extends GCN by bringing attention mechanism






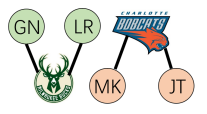
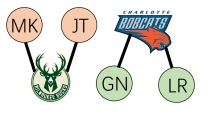
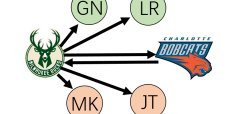
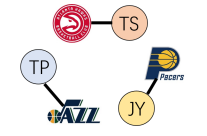
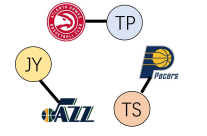
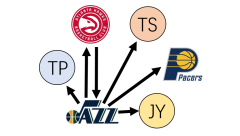
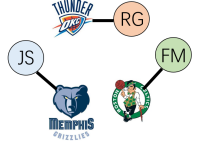
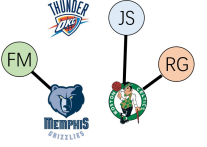
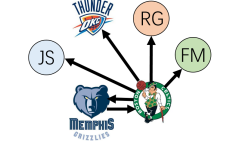
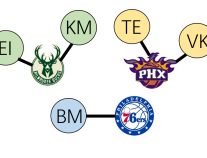

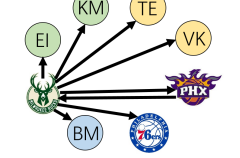
	$G_{text}$	$G'_{text}$	Selected Shortcuts	Results in Text-Subgraph
T1				✓
T2				✓
T3				wrong predictions: (TS, player, Jazz) (TS, teammate, Pacers)
T4				✓
T5				missing triples: (VK, teammate, BM) (TE, teammate, BM)

Table 4: Examples of visualization for the attention on shortcuts. Each row shows an example of one type of trade.  $G_{text}$  and  $G'_{text}$  represent the actual text-subgraphs before and after the trade, respectively. *Selected Shortcuts* represents the shortcuts selected by the attention mechanism, and the arrows indicate the directions of message passing. *Results in Text-Subgraph* lists the prediction errors of GUpdater.

into GNN. To better capture KG information, **R-GCN**(Schlichtkrull et al., 2018) was proposed to incorporate relation embeddings into GNN.

**Relation Extraction** task aims to extract relations from texts. Current relation extraction models are mainly under distant supervision (Mintz et al., 2009), and are trained in bag level. **PCNN** (Zeng et al., 2015) divides the sentence into three pieces and applies max-pooling in a piecewise manner after the convolution layer. **APCNN** (Ji et al., 2017) uses sentence-level attention to select multiple valid sentences with different weights in a bag. (Luo et al., 2017) uses a transition matrix to model the noise and use curriculum for training.

## 6 Conclusion

In this paper, we propose a new text-based KG updating task and construct a dataset, NBAtransactions, for evaluation. We design a novel GNN-based model, GUpdater, which uses text information to guide the message passing through the KG structure. Experiments show that our model can effectively handle both explicit and implicit information, and perform necessary link-adding and

link-deleting operations accordingly. In the future, we will try to investigate how to update KGs when entities are involved in several successive events.

## Acknowledgements

This work is supported in part by the National Hi-Tech R&D Program of China (No. 2018YFC0831900) and the NSFC Grants (No. 61672057, 61672058). For any correspondence, please contact Yansong Feng.

## References

- 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*, pages 2787–2795.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural

- message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.
- Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696.
- Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*.
- Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- Jiaqing Liang, Sheng Zhang, and Yanghua Xiao. 2017. How to keep a knowledge base synchronized with its encyclopedia source.
- Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.
- Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. *arXiv preprint arXiv:1705.03995*.
- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.
- Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.
- Mohamed Morsey, Jens Lehmann, Sören Auer, Claus Stadler, and Sebastian Hellmann. 2012. Dbpedia and the live extraction of structured data from wikipedia. *Program*, 46(2):157–181.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.
- Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and Charles E Leiserson. 2019. Evolvegn: Evolving graph convolutional networks for dynamic graphs. *arXiv preprint arXiv:1902.10191*.
- Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.
- Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2).
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119.
- 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*.
- Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.