
Neural Consciousness Flow

Xiaoran Xu¹, Wei Feng¹, Zhiqing Sun², Zhi-Hong Deng²

¹Hulu LLC, Beijing, China

{xiaoran.xu, wei.feng}@hulu.com

²Peking University, Beijing, China

{1500012783, zhdeng}@pku.edu.cn

Abstract

The ability of reasoning beyond data fitting is substantial to deep learning systems in order to make a leap forward towards artificial general intelligence. A lot of efforts have been made to model neural-based reasoning as an iterative decision-making process based on recurrent networks and reinforcement learning. Instead, inspired by *the consciousness prior* proposed by Yoshua Bengio [1], we explore reasoning with the notion of attentive awareness from a cognitive perspective, and formulate it in the form of attentive message passing on graphs, called **neural consciousness flow (NeuCFlow)**. Aiming to bridge the gap between deep learning systems and reasoning, we propose an attentive computation framework with a three-layer architecture, which consists of an *unconsciousness flow layer*, a *consciousness flow layer*, and an *attention flow layer*. We implement the NeuCFlow model with graph neural networks (GNNs) and conditional transition matrices. Our attentive computation greatly reduces the complexity of vanilla GNN-based methods, capable of running on large-scale graphs. We validate our model for knowledge graph reasoning by solving a series of knowledge base completion (KBC) tasks. The experimental results show NeuCFlow significantly outperforms previous state-of-the-art KBC methods, including the embedding-based and the path-based. The reproducible code can be found by the link¹ below.

1 Introduction

To discover the mystery of consciousness, several competing theories [2, 3, 4, 5] have been proposed by neuroscientists. Despite their contradictory claims, they share a common notion that consciousness is a cognitive state of experiencing one's own existence, i.e. the state of awareness. Here, we do not refer to those elusive and mysterious meanings attributed to the word "consciousness". Instead, we focus on the basic idea, *awareness* or *attentive awareness*, to derive a neural network-based attentive computation framework on graphs, attempting to mimic the phenomenon of consciousness to some extent.

The first work to bring the idea of attentive awareness into deep learning models, as far as we know, is Yoshua Bengio's consciousness prior [1]. He points out the process of disentangling higher-level abstract factors from full underlying representation and forming a low-dimensional combination of a few selected factors or concepts to constitute a conscious thought. Bengio emphasizes the role of attention mechanism in expressing awareness, which helps focus on a few elements of state representation at a given moment and combining them to make a statement, an action or policy. Two recurrent neural networks (RNNs), the representation RNN and the consciousness RNN, are used to summarize the current and recent past information and encode two types of state, the unconscious state denoted by a full high-dimensional vector before applying attention, and the conscious state by a derived low-dimensional vector after applying attention.

¹<https://github.com/netpaladinx/NeuCFlow>

Inspired by the consciousness prior, we develop an attentive message passing mechanism. We model query-dependent states as motivation to drive iterative sparse access to an underlying large graph and navigate information flow via a few nodes to reach a target. Instead of using RNNs, we use two GNNs [6, 7] with node state representations. Nodes sense nearby topological structures by exchanging messages with neighbors, and then use aggregated information to update their states. However, the standard message passing runs globally and uniformly. Messages gathered by a node can come from possibly everywhere and get further entangled by aggregation operations. Therefore, we need to draw a query-dependent or context-aware local subgraph to guide message passing. Nodes within such a subgraph are densely connected, forming a community to further exchange and share information, reaching some resonance, and making subsequent decisions collectively to expand the subgraph and navigate information flow. To support such attentive information flow, we design an *attention flow layer* above two GNNs. One GNN uses the standard message passing over a full graph, called *unconsciousness flow layer*, while the other GNN runs on a subgraph built by attention flow, called *consciousness flow layer*. These three flow layers constitute our attentive computation framework.

We realize the connection between attentive awareness and reasoning. A reasoning process is understood as a sequence of obvious or interpretable steps, either deductive, inductive, or abductive, to derive a less obvious conclusion. From the aspect of awareness, reasoning requires computation to be self-attentive or self-aware during processing in a way different from fitting by a black box. Therefore, interpretability must be one of the properties of reasoning. Taking KBC tasks as an example, many embedding-based models [8, 9, 10, 11, 12, 13] can do a really good job in link prediction, but lacking interpretation makes it hard to argue for their reasoning ability. People who aim at knowledge graph reasoning mainly focus on the path-based models using RL [14, 15, 16, 17] or logic-like methods [18, 19] to explicitly model a reasoning process to provide interpretations beyond predictions. Here, instead, we apply a flow-based attention mechanism, proposed in [20], as an alternative to RL for learning composition structure. In a manner of flowing, attention can propagate to cover a broader scope and increase the chance to hit a target. It maintains an end-to-end differentiable style, contrary to the way RL agents learn to choose a discrete action.

Other crucial properties of reasoning include relational inductive biases and iterative processing. Therefore, GNNs [6, 7] are a better choice compared to RNNs for encoding structured knowledge explicitly. Compared with the majority of previous GNN literature, focusing on the computation side, making neural-based architectures more composable and complex, we put a cognitive insight into it under the notion of attentive awareness. Specifically, we design an attention flow layer to chain attention operations directly with transition matrices, parallel to the message-passing pipeline to get less entangled with representation computation. This gives our model the ability to select edges step by step during computation and attend to a query-dependent subgraph, making a sharper prediction due to the disentanglement. These extracted subgraphs can reduce the computation cost greatly. In practice, we find our model can be applied to very large graphs with millions of nodes, such as the YAGO3-10 dataset, even running on a single laptop.

Our contributions are three-fold: (1) We propose an attentive computation framework on graphs, combining GNNs’ representation power with explicit reasoning pattern, motivated by the cognitive notion of attentive awareness. (2) We exploit query-dependent subgraph structure, extracted by an attention flow mechanism, to address two shortcomings of most GNN implementations: the complexity and the non-context-aware aggregation schema. (3) We design a specific architecture for KBC tasks and demonstrate our model’s strong reasoning capability compared to the state of the art, showing that a compact query-dependent subgraph is better than a path as a reasoning pattern.

2 Related Work

KBC and knowledge graph reasoning. Early work for KBC, including TransE [8] and its analogues [21, 22, 23], DistMult [9], ConvE [10] and ComplEx [11], focuses on learning embeddings of entities and relations. Some recent work of this line [12, 13] achieves high accuracy, yet unable to explicitly deal with compositional relationships that is crucial for reasoning. Another line aims to learn inference paths [14, 24, 25, 26, 27, 28] for knowledge graph reasoning, such as DeepPath [15], MINERVA [16], and M-Walk [17], using RL to learn multi-hop relational paths over a graph towards a target given a query. However, these approaches, based on policy gradients or Monte Carlo tree search, often suffer from low sample efficiency and sparse rewards, requiring a large number of rollouts or

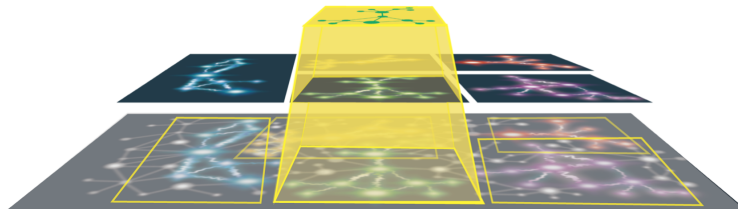


Figure 1: Illustration for the three-layer attentive computation framework. The bottom is a unified *unconsciousness flow layer*, the middle contains small disentangled subgraphs to run attentive message passing separately, constituting a *consciousness flow layer*, and the top is an *attention flow layer* for extracting local subgraph structures.

running many simulations, and also the sophisticated reward function design. Other efforts include learning soft logical rules [18, 19] or compositional programs [29] to reason over knowledge graphs.

Relational reasoning by GNNs and attention mechanisms. Relational reasoning is regarded as the key component of humans’ capacity for combinatorial generalization, taking the form of entity- and relation-centric organization to reason about the composition structure of the world [30, 31, 32, 33, 34]. A multitude of recent implementations [7] encode relational inductive biases into neural networks to exploit graph-structured representation, including graph convolution networks (GCNs) [35, 36, 37, 38, 39, 40, 41, 42] and graph neural networks [6, 43, 44, 45, 46], and overcome the difficulty to achieve relational reasoning for traditional deep learning models. These approaches have been widely applied to accomplishing real-world reasoning tasks (such as physical reasoning [45, 47, 48, 49, 50, 51], visual reasoning [44, 51, 52, 53, 54], textual reasoning [44, 55, 56], knowledge graph reasoning [41, 57, 58], multiagent relationship reasoning [59, 60], and chemical reasoning [46]), solving algorithmic problems (such as program verification [43, 61], combinatorial optimization [62, 63, 64], state transitions [65], and boolean satisfiability [66]), or facilitating reinforcement learning with the structured reasoning or planning ability [67, 68, 49, 50, 69, 70, 71]. Variants of GNN architectures have been developed with different focuses. Relation networks [44] use a simple but effective neural module to equip deep learning models with the relational reasoning ability, and its recurrent versions [55, 56] do multi-step relational inference for long periods; Interaction networks [45] provide a general-purpose learnable physics engine, and two of its variants are visual interaction networks [51] learning directly from raw visual data, and vertex attention interaction networks [60] with an attention mechanism; Message passing neural networks [46] unify various GCNs and GCNs into a general message passing formalism by analogy to the one in graphical models.

Despite the strong representation power of GNNs, recent work points out its drawbacks that limit its capability. The vanilla message passing or neighborhood aggregation schema cannot adapt to strongly diverse local subgraph structure, causing performance degeneration when applying a deeper version or running more iterations [72], since a walk of more steps might drift away from local neighborhood with information washed out via averaging. It is suggested that covariance rather than invariance to permutations of nodes and edges is preferable [73], since being fully invariant by summing or averaging messages may worsen the representation power, lacking steerability. In this context, our model expresses permutation invariance under a constrained compositional transformation according to the group of possible permutations within each extracted query-dependent subgraph rather than the underlying full graph. Another drawback is the heavy computation complexity. GNNs are notorious for its poor scalability due to its quadratic complexity in the number of nodes when graphs are fully connected. Even scaling linearly with the number of edges by exploiting structure sparsity can still cause trouble on very large graphs, making selective or attentive computation on graphs so desirable.

Neighborhood attention operation can alleviate some limitation on GNNs’ representation power by specifying different weights to different nodes or nodes’ features [74, 60, 53, 75]. These approaches often use multi-head self-attention to focus on specific interactions with neighbors when aggregating messages, inspired by [76, 77, 78] originally for capturing long range dependencies. We notice that most graph-based attention mechanisms attend over neighborhood in a single-hop fashion, and [60] claims that the multi-hop architecture does not help in experiments, though they expect multiple hops to offer the potential to model high-order interaction. However, a flow-based design of attention in [20] shows a promising way to characterize long distance dependencies over graphs, breaking the isolation of attention operations and stringing them in chronological order by transition matrices, like the spread of a random walk, parallel to the message-passing pipeline.

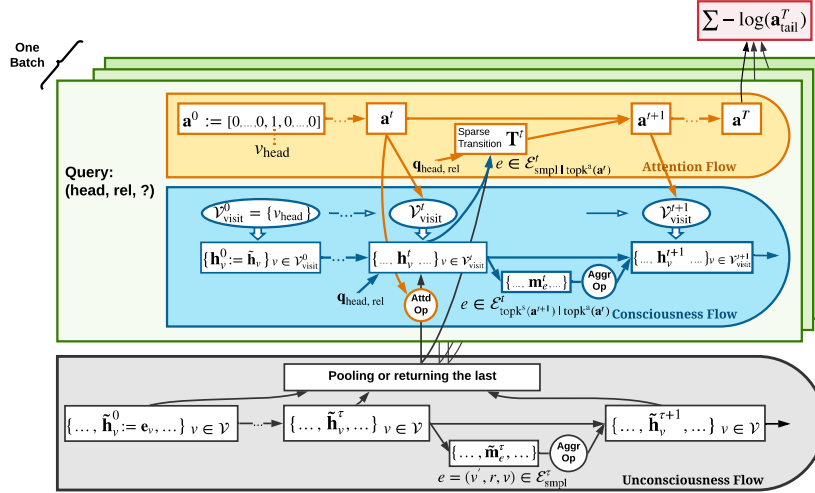


Figure 2: The neural consciousness flow architecture.

It is natural to extend relational reasoning to graph structure inference or graph generation, such as reasoning about a latent interaction graph explicitly to acquire knowledge of observed dynamics [48], or learning generative models of graphs [79, 80, 81, 82]. Soft plus hard attention mechanisms may be a better alternative to probabilistic models that is hard to train with latent discrete variables or might degenerate multi-step predictions due to the inaccuracy (biased gradients) of back-propagation.

3 NeuCFlow Model

3.1 Attentive computation framework

We extend Bengio’s consciousness prior to graph-structured representation. Conscious thoughts are modeled by a few selected nodes and their edges, forming a context-aware subgraph, cohesive with sharper semantics, disentangled from the full graph. The underlying full graph forms the initial representation, entangled but rich, to help shape potential high-level subgraphs. We use attention flow to navigate conscious thoughts, capturing a step-by-step reasoning pattern. The attentive computation framework, as illustrated in Figure 1, consists of: (1) an *unconsciousness flow* (*U-Flow*) layer, (2) a *consciousness flow* (*C-Flow*) layer, and (3) an *attention flow* (*A-Flow*) layer, with four guidelines to design a specific implementation as follows:

- *U-Flow* corresponds to a low-level computation graph for full state representation learning.
- *C-Flow* contains high-level disentangled subgraphs for context-aware representation learning.
- *A-Flow* is conditioned by both *U-Flow* and *C-Flow*, and also motivate *C-Flow* but not *U-Flow*.
- Information can be accessed by *C-Flow* from *U-Flow* with the help of *A-Flow*.

3.2 Model architecture design for knowledge graph reasoning

We choose KBC tasks to do KG reasoning. We let $(\mathcal{V}, \mathcal{E})$ denote a KG where \mathcal{V} is a set of nodes (or entities) and \mathcal{E} is a set of edges (or relations). A KG is viewed as a directed graph with each edge represented by a triple $\langle head, rel, tail \rangle$, where *head* is the head entity, *tail* is the tail entity, and *rel* is their relation type. The aim of a KBC task is to predict potential unknown links, i.e., which entity is likely to be the tail given a query $\langle head, rel, ? \rangle$ with the head and the relation type specified.

The model architecture has three core components as shown in Figure 2. We here use the term "component" instead of "layer" to differentiate our flow layers from the referring normally used in neural networks, as each flow layer is more like a block containing many neural network layers.

U-Flow component. We implement this component over the full graph using the standard message passing mechanism [46]. If the graph has an extremely large number of edges, we sample a subset

of edges, $\mathcal{E}_{\text{smp1}}^\tau \subset \mathcal{E}$, randomly each step when running message passing. For each batch of input queries, we let the representation computed by the U-Flow component be shared across these different queries, which means U-Flow is query-independent, with its state representation tensors containing no batch dimension, so that its complexity does not scale with the batch size and the saved computation resources can be allocated to sampling more edges. In U-Flow, each node v has a learnable embedding \mathbf{e}_v and a dynamical state $\tilde{\mathbf{h}}_v^\tau$ for step τ , called unconscious node states, where the initial $\tilde{\mathbf{h}}_v^0 := \mathbf{e}_v$ for all $v \in \mathcal{V}$. Each edge type r also has a learnable embedding \mathbf{e}_r , and edge $\langle v', r, v \rangle$ can produce a message, denoted by $\tilde{\mathbf{m}}_{\langle v', r, v \rangle}^\tau$, at step τ . The U-Flow component includes:

- Message function: $\tilde{\mathbf{m}}_{\langle v', r, v \rangle}^\tau = \psi_{\text{unc}}(\tilde{\mathbf{h}}_{v'}^\tau, \mathbf{e}_r, \tilde{\mathbf{h}}_v^\tau)$, where $\langle v', r, v \rangle \in \mathcal{E}_{\text{smp1}}^\tau$.
- Message aggregation: $\tilde{\mu}_v^\tau = \frac{1}{\sqrt{N_v^\tau}} \sum_{v', r} \tilde{\mathbf{m}}_{\langle v', r, v \rangle}^\tau$, where $\langle v', r, v \rangle \in \mathcal{E}_{\text{smp1}}^\tau$.
- Node state update function: $\tilde{\mathbf{h}}_v^{\tau+1} = \tilde{\mathbf{h}}_v^\tau + \delta_{\text{unc}}(\tilde{\mu}_v^\tau, \tilde{\mathbf{h}}_v^\tau, \mathbf{e}_v)$, where $v \in \mathcal{V}$.

We compute messages only for the sampled edges, $\langle v', r, v \rangle \in \mathcal{E}_{\text{smp1}}^\tau$, each step. Functions ψ_{unc} and δ_{unc} are implemented by a two-layer MLP (using leakyReLU for the first layer and tanh for the second layer) with input arguments concatenated respectively. Messages are aggregated by dividing the sum by the square root of the number of sampled neighbors that send messages, preserving the scale of variance. We use a residual adding to update each node state instead of a GRU or a LSTM. After running U-Flow for \mathcal{T} steps, we return a pooling result or simply the last, $\tilde{\mathbf{h}}_v := \tilde{\mathbf{h}}_v^\mathcal{T}$, to feed into downstream components.

C-Flow component. C-Flow is query-dependent, which means that conscious node states, denoted by \mathbf{h}_v^t , have a batch dimension representing different input queries, making the complexity scale with the batch size. However, as C-Flow uses attentive message passing, running on small local subgraphs each conditioned by a query, we leverage the sparsity to record \mathbf{h}_v^t only for the visited nodes $v \in \mathcal{V}_{\text{visit}}^t$. For example, when $t = 0$, for query $\langle \text{head}, \text{rel}, ? \rangle$, we start from node head , with $\mathcal{V}_{\text{visit}}^0 = \{\text{head}\}$ being a singleton, and thus record $\mathbf{h}_{\text{head}}^0$ only. When computing messages, denoted by $\mathbf{m}_{\langle v', r, v \rangle}^t$, in C-Flow, we use a sampling-attending procedure, explained in Section 3.3, to further control the number of computed edges. The C-Flow component has:

- Message function: $\mathbf{m}_{\langle v', r, v \rangle}^t = \psi_{\text{con}}(\mathbf{h}_{v'}^t, \mathbf{c}_r, \mathbf{h}_v^t)$, where $\langle v', r, v \rangle \in \mathcal{E}_{\text{topk}^s(\mathbf{a}^{t+1}) | \text{topk}^a(\mathbf{a}^t)}$, and $\mathbf{c}_r = [\mathbf{e}_r, \mathbf{q}_{\text{head}}, \mathbf{q}_{\text{rel}}]$.
- Message aggregation: $\mu_v^t = \frac{1}{\sqrt{N_v^t}} \sum_{v', r} \mathbf{m}_{\langle v', r, v \rangle}^t$, where $\langle v', r, v \rangle \in \mathcal{E}_{\text{topk}^s(\mathbf{a}^{t+1}) | \text{topk}^a(\mathbf{a}^t)}$.
- Node state attending function: $\tilde{\eta}_v^{t+1} = a_v^{t+1} \mathbf{A} \cdot \tilde{\mathbf{h}}_v$, where $a_v^{t+1} = \mathbf{a}^{t+1}[v]$ and $v \in \mathcal{V}_{\text{visit}}^{t+1}$.
- Node state update function: $\mathbf{h}_v^{t+1} = \mathbf{h}_v^t + \delta_{\text{con}}(\mu_v^t, \mathbf{h}_v^t, \mathbf{c}_v^{t+1})$, where $\mathbf{c}_v^{t+1} = [\tilde{\eta}_v^{t+1}, \mathbf{q}_{\text{head}}, \mathbf{q}_{\text{rel}}]$.

C-Flow and U-Flow share the embeddings \mathbf{e}_r . A query is represented by its head and relation embeddings, $\mathbf{q}_{\text{head}} := \mathbf{e}_{\text{head}}$ and $\mathbf{q}_{\text{rel}} := \mathbf{e}_{\text{rel}}$, participating in computing messages and updating node states. We here select a subset of edges, $\mathcal{E}_{\text{topk}^s(\mathbf{a}^{t+1}) | \text{topk}^a(\mathbf{a}^t)}$, rather than sampling, according to edges between the attended nodes at step t and the seen nodes at step $t + 1$, defined in Section 3.3, as shown in Figure 3. We introduce the node state attending function to pass an unconscious state $\tilde{\mathbf{h}}_v$ to C-Flow adjusted by a scalar attention a_v^{t+1} and a learnable matrix \mathbf{A} . We initialize $\mathbf{h}_v^0 := \tilde{\mathbf{h}}_v$ for $v \in \mathcal{V}_{\text{visit}}^0$, treating the rest as zero states.

A-Flow component. Attention flow is represented by a series of probability distributions changing across steps, denoted as $\mathbf{a}^t, t = 1, 2, \dots, T$. The initial distribution \mathbf{a}^0 is a one-hot vector with $\mathbf{a}^0[\text{head}] = 1$. To spread attention, we need to compute transition matrices \mathbf{T}^t each step. Given that A-Flow is conditioned by both U-Flow and C-Flow, we model the transition from v' to v by two types of interaction: conscious-to-conscious, $\mathbf{h}_{v'}^t \sim \mathbf{h}_v^t$, and conscious-to-unconscious, $\mathbf{h}_{v'}^t \sim \tilde{\mathbf{h}}_v$. The former favors previously visited nodes, while the latter is useful to attend to unseen nodes.

$$\mathbf{T}^t[:, v'] = \text{softmax}_{v \in \mathcal{N}_{v'}^t} \left(\sum_r \alpha_{\text{cc}}(\mathbf{h}_{v'}^t, \mathbf{c}_r, \mathbf{h}_v^t) + \sum_r \alpha_{\text{cu}}(\mathbf{h}_{v'}^t, \mathbf{c}_r, \tilde{\mathbf{h}}_v) \right)$$

where $\alpha_{\text{cc}} = \text{MLP}(\mathbf{h}_{v'}^t, \mathbf{c}_r)^T \Theta_{\text{cc}} \text{MLP}(\mathbf{h}_v^t, \mathbf{c}_r)$ and $\alpha_{\text{cu}} = \text{MLP}(\mathbf{h}_{v'}^t, \mathbf{c}_r)^T \Theta_{\text{cu}} \text{MLP}(\tilde{\mathbf{h}}_v, \mathbf{c}_r)$, and Θ_{cc} and Θ_{cu} are two learnable matrices. Each MLP uses one single layer with the leakyReLU

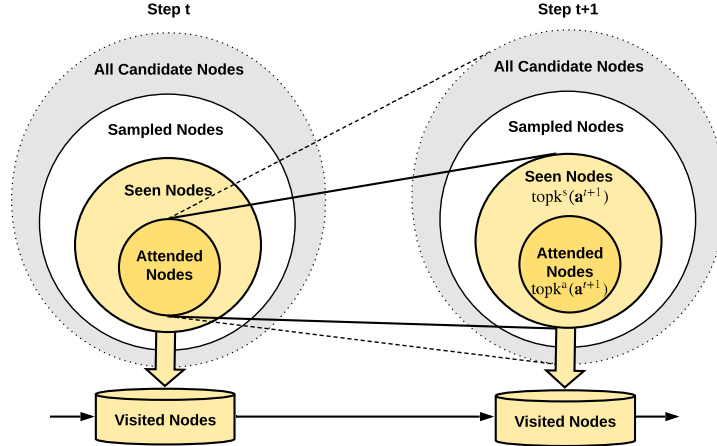


Figure 3: The iterative sampling-attending procedure for attentive complexity reduction, balancing the coverage as well as the complexity.

activation. To reduce the complexity for computing \mathbf{T}^t , we select attended nodes, $v' \in \text{topk}^a(\mathbf{a}^t)$, which is the set of nodes with the k -largest attention, and then sample v from v' neighbors as next nodes. Then, we compute a sparse \mathbf{T}^t according to edges $\langle v', r, v \rangle \in \mathcal{E}_{\text{smp}| \text{topk}^a(\mathbf{a}^t)}$. Due to the fact that the attended nodes may not carry all attention, a small amount of attention can be lost during transition, causing the total amount to decrease. Therefore, we use a renormalized version, $\mathbf{a}^{t+1} = \mathbf{T}^t \mathbf{a}^t / \|\mathbf{T}^t \mathbf{a}^t\|$. We use the final attention on the tail as the probability for prediction to compute the training objective, as shown in Figure 2.

3.3 Complexity reduction by iterative sampling and attending

Previously, we use edge sampling, in a globally and uniformly random manner, to address the complexity issue in U-Flow, where we are not concerned about the batch size. Here, we need to confront the complexity that scales with the batch size in C-Flow. Suppose that we run a normal message passing for T steps on a KG with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges for a batch of N queries. Then, the complexity is $\mathcal{O}(NTD(|\mathcal{V}| + |\mathcal{E}|))$ where D represents the number of representation dimensions. The complexity can be reduced to $\mathcal{O}(NTD(|\mathcal{V}| + |\mathcal{E}_{\text{smp}}|))$ by using edges sampling. T is a small positive integer, often less than 10. D is normally between 50 and 200, and being too small for D would lead to underfitting. In U-Flow, we have $N = 1$, while in C-Flow, let us say $N = 100$. Then, to maintain the same complexity as U-Flow, we have to reduce the sampling rate by a factor of 100 on each query. However, the U-Flow's edge sampling procedure is for the full graph, and it is inappropriate to apply to C-Flow on each query due to the reduced sample rate. Also, when $|\mathcal{V}|$ becomes as large as $|\mathcal{E}_{\text{smp}}|$, we also need to consider decreasing $|\mathcal{V}|$.

Good news is that C-Flow deals with a local subgraph for each query so that we only record a few selected nodes, called *visited nodes*, denoted by $\mathcal{V}_{\text{visit}}^t$. We can see that $|\mathcal{V}_{\text{visit}}^t|$ is much less than $|\mathcal{V}|$. The initial $\mathcal{V}_{\text{visit}}^0$, when $t = 0$, contains only one node v_{head} , and then $\mathcal{V}_{\text{visit}}^t$ is enlarged each step by adding new nodes during spreading. When propagating messages, we only care about the one-step neighborhood each step. However, the spreading goes so rapidly that after only a few steps it covers almost all nodes, causing the number of computed edges to increase dramatically. The key to address the problem is that we need to constrain the scope of nodes we jump from each step, i.e., the core nodes that determine where we can go based on where we depart from. We call them *attended nodes*, which are in charge of the attending-from horizon, selected by $\text{topk}^a(\mathbf{a}^t)$ based on the current attention \mathbf{a}^t . Given the set of attended nodes, we still need edge sampling over their neighborhoods in case of a hub node of extremely high degree. Here, we face a tricky problem that is to make a trade-off between the coverage and the complexity when sampling over the neighborhoods. Also, we need to well maintain these coherent context-aware node states and avoid possible noises or drifting away caused by sampling neighbors randomly. Therefore, we introduce an attending-to horizon inside the sampling horizon. We compute A-Flow over the sampling horizon with a smaller dimension to compute the attention, exchanged for sampling more neighbors to increase the coverage. Based

Table 1: Statistics of the six KG datasets. A KG is built on all training triples including their inverse triples. Note that we do not count the inverse triples in FB15K, FB15K-237, WN18, WN18RR, and YAGO3-10 as shown below to be consistent with the statistics reported in other papers, though we include them in the training, validation and test set. PME (tr) means the proportion of multi-edge triples in train; PME (te) means the proportion of multi-edge triples in test; AvgD (te) means the average length of shortest paths connecting each head-tail pair in test.

Dataset	#Entities	#Rels	#Train	#Valid	#Test	PME (tr)	PME (te)	AvgD (te)
FB15K	14,951	1,345	483,142	50,000	59,071	81.2%	80.9%	1.22
FB15K-237	14,541	237	272,115	17,535	20,466	38.0%	0%	2.25
WN18	40,943	18	141,442	5,000	5,000	93.1%	94.0%	1.18
WN18RR	40,943	11	86,835	3,034	3,134	34.5%	35.0%	2.87
NELL995	74,536	200	149,678	543	2,818	100%	41.0%	2.06
YAGO3-10	123,188	37	1,079,040	5,000	5,000	56.4%	56.0%	1.75

Table 2: Comparison results on the FB15K-237 and WN18RR datasets. Results of [♠] are taken from [83], results of [♣] from [10], results of [♥] from [17], results of [◇] from [12], and results of [△] from [16]. Some collected results only have a metric score while some including ours take the form of "mean (standard deviation)".

Metric (%)	FB15K-237				WN18RR			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
TransE [♠]	-	-	46.5	29.4	-	-	50.1	22.6
DistMult [♣]	15.5	26.3	41.9	24.1	39	44	49	43
DistMult [♥]	20.6 (.4)	31.8 (.2)	-	29.0 (.2)	38.4 (.4)	42.4 (.3)	-	41.3 (.3)
CompLex [♣]	15.8	27.5	42.8	24.7	41	46	51	44
CompLex [♥]	20.8 (.2)	32.6 (.5)	-	29.6 (.2)	38.5 (.3)	43.9 (.3)	-	42.2 (.2)
ConvE [♣]	23.7	35.6	50.1	32.5	40	44	52	43
ConvE [♥]	23.3 (.4)	33.8 (.3)	-	30.8 (.2)	39.6 (.3)	44.7 (.2)	-	43.3 (.2)
RotatE [◇]	24.1	37.5	53.3	33.8	42.8	49.2	57.1	47.6
NeuralLP [♥]	18.2 (.6)	27.2 (.3)	-	24.9 (.2)	37.2 (.1)	43.4 (.1)	-	43.5 (.1)
MINERVA [♥]	14.1 (.2)	23.2 (.4)	-	20.5 (.3)	35.1 (.1)	44.5 (.4)	-	40.9 (.1)
MINERVA [△]	-	-	45.6	-	41.3	45.6	51.3	-
M-Walk [♥]	16.5 (.3)	24.3 (.2)	-	23.2 (.2)	41.4 (.1)	44.5 (.2)	-	43.7 (.1)
NeuCFlow	28.6 (.1)	40.3 (.1)	53.0 (.3)	36.9 (.1)	44.4 (.4)	49.7 (.8)	55.8 (.5)	48.2 (.5)

on the newly computed attention \mathbf{a}^{t+1} , we select a smaller subset of nodes, $\text{topk}^s(\mathbf{a}^{t+1})$, to receive messages in C-Flow, called *seen nodes*, in charge of the attending-to horizon. The next attending-from horizon is chosen by $\text{topk}^a(\mathbf{a}^{t+1}) \subset \text{topk}^s(\mathbf{a}^{t+1})$, a sub-horizon of the current attending-to horizon. All seen and attended nodes are stored as visited nodes along steps. We illustrate this sampling-attending procedure in Figure 3.

To compute our reduced complexity, we let N_e be the maximum number of sampled edges per attended node per step, N_s the maximum number of seen nodes per step, and N_a the maximum number of attended nodes per step. We also denote the dimension number used in A-Flow as D_a . For one batch, the complexity of C-Flow is $\mathcal{O}(NTD(N_a + N_s + N_a N_s))$ for the worst case, where attended and seen nodes are fully connected, and $\mathcal{O}(NTD \cdot c(N_a + N_s))$ in most cases, where c is a small constant. The complexity of A-Flow is $\mathcal{O}(NTD_a N_a N_e)$ where D_a is much smaller than D .

4 Experiments

4.1 Datasets and experimental settings

Datasets. We evaluate our model using six large KG datasets²: FB15K, FB15K-237, WN18, WN18RR, NELL995, and YAGO3-10. FB15K-237 [84] is sampled from FB15K [8] with redundant relations removed, and WN18RR [10] is a subset of WN18 [8] removing triples that cause test leakage. Thus, they are both considered more challenging. NELL995 [15] has separate datasets

²<https://github.com/netpaladinx/NeuCFlow/tree/master/data>

Table 3: Comparison results on the FB15K and WN18 datasets. Results of [♠] are taken from [86], results of [♣] are from [10], results of [◇] are from [12], and results of [♥] are from [19]. Our results take the form of "mean (standard deviation)".

Metric (%)	FB15K				WN18			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
TransE [♠]	29.7	57.8	74.9	46.3	11.3	88.8	94.3	49.5
HolE [♠]	40.2	61.3	73.9	52.4	93.0	94.5	94.9	93.8
DistMult [♣]	54.6	73.3	82.4	65.4	72.8	91.4	93.6	82.2
ComplEx [♣]	59.9	75.9	84.0	69.2	93.6	93.6	94.7	94.1
ConvE [♣]	55.8	72.3	83.1	65.7	93.5	94.6	95.6	94.3
RotatE [◇]	74.6	83.0	88.4	79.7	94.4	95.2	95.9	94.9
NeuralLP [♥]	-	-	83.7	76	-	-	94.5	94
NeuCFlow	72.6 (.4)	78.4 (.4)	83.4 (.5)	76.4 (.4)	91.6 (.8)	93.6 (.4)	94.9 (.4)	92.8 (.6)

Table 4: Comparison results on the YAGO3-10 dataset. Results of [♠] are taken from [10], and results of [♣] are from [13].

Metric (%)	YAGO3-10			
	H@1	H@3	H@10	MRR
DistMult [♠]	24	38	54	34
ComplEx [♠]	26	40	55	36
ConvE [♠]	35	49	62	44
ComplEx-N3 [♣]	-	-	71	58
NeuCFlow	48.4	59.5	67.9	55.3

for 12 query relations each corresponding to a single-query-relation KBC task. YAGO3-10 [85] contains the largest KG with millions of edges. Their statistics are shown in Table 1. We find some statistical differences between train and test. In a KG with all training triples as its edges, a triple (*head, rel, tail*) is considered as a multi-edge triple if the KG contains other triples that also connect *head* and *tail* ignoring the direction. We notice that FB15K-237 is a special case compared with the others, as there are no edges in its KG directly linking any pair of *head* and *tail* in test. Therefore, when using training triples as queries to train our model, given a batch, for FB15K-237, we cut off from the KG all triples connecting the head-tail pairs in the given batch, ignoring relation types and edge directions, forcing the model to learn a composite reasoning pattern rather than a single-hop pattern, and for the rest datasets, we only remove the triples of this batch and their inverse from the KG before training on this batch.

Experimental settings. We use the same data split protocol as in many papers [10, 15, 16]. We create a KG, a directed graph, consisting of all train triples and their inverse added for each dataset except NELL995, since it already includes reciprocal relations. Besides, every node in KGs has a self-loop edge to itself. We also add inverse relations into the validation and test set to evaluate the two directions. For evaluation metrics, we use HITS@1,3,10 and the mean reciprocal rank (MRR) in the filtered setting for FB15K-237, WN18RR, FB15K, WN18, and YAGO3-10, and use the mean average precision (MAP) for NELL995’s single-query-relation KBC tasks. For NELL995, we follow the same evaluation procedure as in [15, 16, 17], ranking the answer entities against the negative examples given in their experiments. We run our experiments using a 12G-memory GPU, TITAN X (Pascal), with Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz. Our code is written in Python based on TensorFlow 2.0 and NumPy 1.16.

4.2 Baselines and comparison results

Baselines. We compare our model against embedding-based approaches, including TransE [8], TransR [22], DistMult [9], ConvE [10], ComplE [11], HolE [86], RotatE [12], and ComplEx-N3 [13], and path-based approaches that use RL methods, including DeepPath [15], MINERVA [16], and M-Walk [17], and also that uses learned neural logic, NeuralLP [19]. For all the baselines, we quote the results from the corresponding papers instead of rerunning them. For our method, we run

Table 5: Comparison results of MAP scores (%) on NELL995’s single-query-relation KBC tasks. We take our baselines’ results from [17]. All results take the form of "mean (standard deviation)" except for TransE and TransR.

Tasks	NeuCFlow	M-Walk	MINERVA	DeepPath	TransE	TransR
AthletePlaysForTeam	83.9 (0.5)	84.7 (1.3)	82.7 (0.8)	72.1 (1.2)	62.7	67.3
AthletePlaysInLeague	97.5 (0.1)	97.8 (0.2)	95.2 (0.8)	92.7 (5.3)	77.3	91.2
AthleteHomeStadium	93.6 (0.1)	91.9 (0.1)	92.8 (0.1)	84.6 (0.8)	71.8	72.2
AthletePlaysSport	98.6 (0.0)	98.3 (0.1)	98.6 (0.1)	91.7 (4.1)	87.6	96.3
TeamPlayssport	90.4 (0.4)	88.4 (1.8)	87.5 (0.5)	69.6 (6.7)	76.1	81.4
OrgHeadQuarteredInCity	94.7 (0.3)	95.0 (0.7)	94.5 (0.3)	79.0 (0.0)	62.0	65.7
WorksFor	86.8 (0.0)	84.2 (0.6)	82.7 (0.5)	69.9 (0.3)	67.7	69.2
PersonBornInLocation	84.1 (0.5)	81.2 (0.0)	78.2 (0.0)	75.5 (0.5)	71.2	81.2
PersonLeadsOrg	88.4 (0.1)	88.8 (0.5)	83.0 (2.6)	79.0 (1.0)	75.1	77.2
OrgHiredPerson	84.7 (0.8)	88.8 (0.6)	87.0 (0.3)	73.8 (1.9)	71.9	73.7
AgentBelongsToOrg	89.3 (1.2)	-	-	-	-	-
TeamPlaysInLeague	97.2 (0.3)	-	-	-	-	-

the experiments three times in each hyperparameter setting on each dataset to report the means and standard deviations of the results. We put the details of our hyperparameter settings in the appendix.

Comparison results and analysis. We first report the comparison on FB15K-23 and WN18RR in Table 2. NeuCFlow has a surprisingly good result, significantly outperforming all the compared methods in HITS@1,3 and MRR on both the two datasets. Compared to the best baseline, RotatE, published very recently, we only lose a few points in HITS@10 but gain a lot in HITS@1,3 and MRR. Based on the observation that NeuCFlow gains a larger amount of advantage when k in HITS@k gets smaller, we speculate that the reasoning ability acquired by NeuCFlow is to make a sharper prediction by exploiting graph-structured composition locally and conditionally, in contrast to embedding-based methods, which totally rely on vectorized representation. When a target becomes too vague to predict, reasoning may lose its great advantage, though still very competitive. However, path-based baselines, with a certain ability to do KG reasoning, perform worse than we expect. We argue that it is inappropriate to think of reasoning, a sequential decision process, as a sequence of nodes, i.e. a path, in KGs. The average length of the shortest paths between heads and tails in the test set in a KG, as shown in Table 1, suggests an extremely short path, making the motivation for using a path pattern almost pointless. The iterative reasoning pattern should be characterized in the form of dynamically varying local graph-structured patterns, holding a bunch of nodes resonating with each other to produce a decision collectively. Then, we run our model on larger KGs, including FB15K, WN18, and YAGO3-10, and summarize the comparison in Table 3,4, where NeuCFlow beats most well-known baselines and achieves a very competitive position against the best state-of-the-art methods. Moreover, we summarize the comparison on NELL995’s tasks in Table 5. NeuCFlow performs the best on five tasks, also being very competitive against M-Walk, the best path-based method as far as we know, on the rest. We find no reporting on the last two tasks from the corresponding papers.

4.3 Experimental analysis

Convergence analysis. During training we find that NeuCFlow converges surprisingly fast. We may use half of training examples to get the model well trained and generalize it to the test, sometimes producing an even better metric score than trained for a full epoch, as shown in Figure 4(A). Compared with the less expensive computation using embedding-based models, although our model takes a large number of edges to compute for each input query, consuming more time on one batch, it does not need a second epoch or even taking all training triples as queries in one epoch, thus saving a lot of training time. The reason may be that all queries are directly from the KG’s edge set and some of them have probably been exploited to construct subgraphs for many times during the training of other queries, so that we might not have to train the model on each query explicitly as long as we have other ways to exploit them.

Component analysis. If we do not run U-Flow, then the unconscious state $\tilde{\mathbf{h}}_v$ is just the initial embedding of node v , and we can still run C-Flow as usual. We want to know whether the U-Flow component is actually useful. Considering that long-distance message passing might bring in less

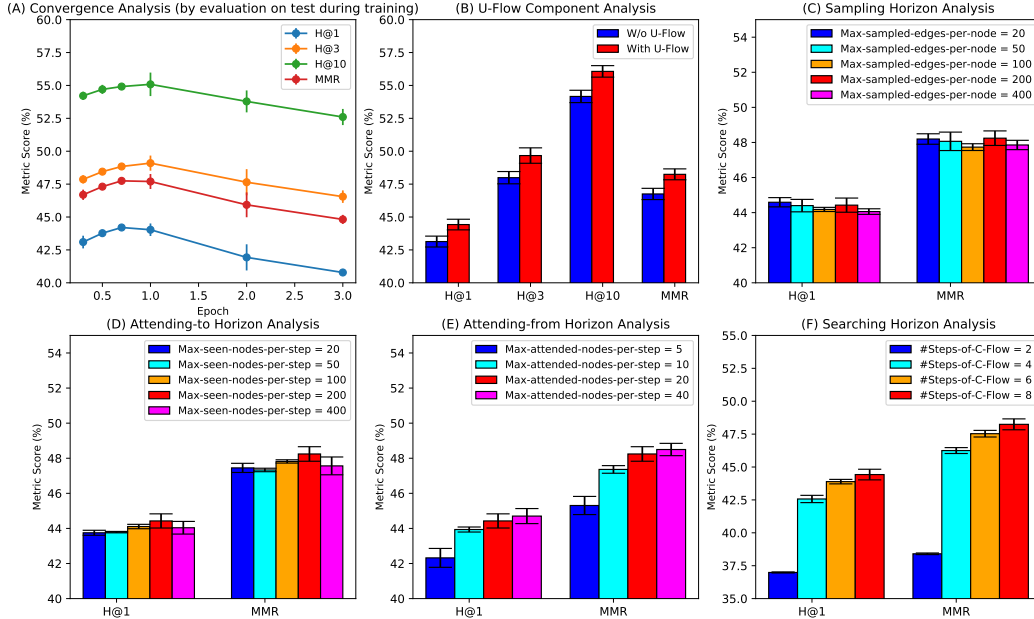


Figure 4: Experimental analysis on WN18RR: (A) During training we pick six model snapshots at time points of 0.3, 0.5, 0.7, 1, 2, and 3 epochs and evaluate them on test; (B) The *w/o U-Flow* uses zero step to run U-Flow, while the *with U-Flow* uses two steps; (C)-(F) are for the sampling, attending and searching horizon analysis based on the standard hyperparameter settings listed in the appendix. The experimental analysis charts on FB15K-237 can be found in the appendix.

informative features, we compare running U-Flow for two steps against totally shutting it down. The result in Figure 4(B) shows that U-Flow brings a small gain in each metric on WN18RR.

Horizon analysis. The sampling, attending and searching horizons determine how large area the flow can spread over. They impact the computation complexity as well as the performance of the model with different degrees depending on the properties of a dataset. Intuitively, enlarging the probe scope by sampling more, attending more, or searching longer, may increase the chance to hit a target. However, the experimental results in Figure 4(C)(D) show that it is not always the case. In Figure 4(E), we can see that increasing the maximum number of the attending-from nodes, i.e. attended nodes, per step is more important, but our GPU does not allow for a larger number to accommodate more intermediate data produced during computation, otherwise causing the error of *ResourceExhaustedError*. Figure 4(F) shows the step number of C-Flow cannot get too small as two.

Attention flow analysis. If attention flow can really capture the way we reason about the world, its process should be conducted in a diverging-converging thinking pattern. Intuitively, first, for the diverging thinking, we search and collect ideas as much as we can; then, for the converging thinking, we try to concentrate our thoughts on one point. To check whether the attention flow has such a pattern, we measure the average entropy of attention distributions varying along steps and also the proportion of attention concentrated at the top-1,3,5 attended nodes. As we expect, attention indeed is more focused at the final step as well as at the beginning.

Time cost analysis. The time cost is affected not only by the scale of a dataset but also by the horizon setting. For each dataset, we list the training time for one epoch corresponding to the standard hyperparameter settings in the appendix. Note that there is always a trade-off between the complexity and the performance. We thus study whether we can reduce the time cost a lot at the price of sacrificing a little performance. We plot the one-epoch training time in Figure 6(A)-(D), using the same settings as we do in the horizon analysis. We can see that *Max-attended-nodes-per-step* and *#Steps-of-C-Flow* affect the training time significantly while *Max-sampled-edges-per-node* and *Max-seen-nodes-per-step* affect very slightly. Therefore, we can use smaller *Max-sampled-edges-per-node* and *Max-seen-nodes-per-step* in order to gain a larger batch size, making the computation more efficiency as shown in Figure 6(E).

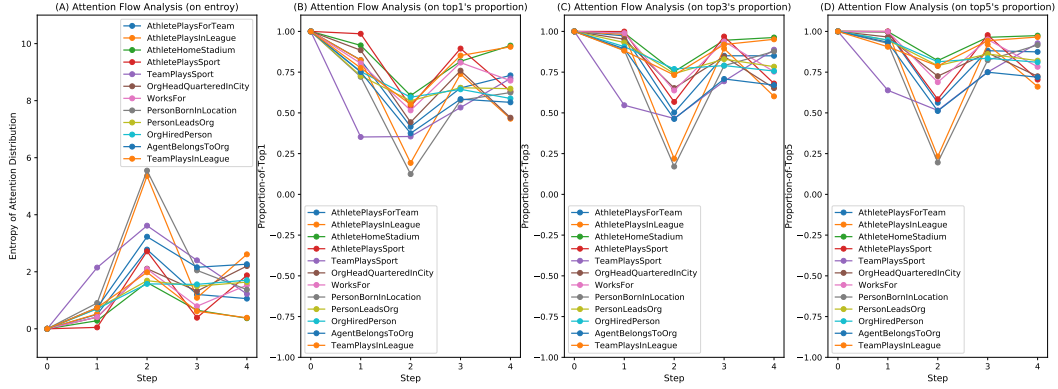


Figure 5: Analysis of attention flow on NELL995 tasks: (A) records how the average entropy of attention distributions varies along steps for each single-query-relation KBC task. (B)(C)(D) measure the changing of the proportion of attention concentrated at the top-1,3,5 attended nodes per step for each task.

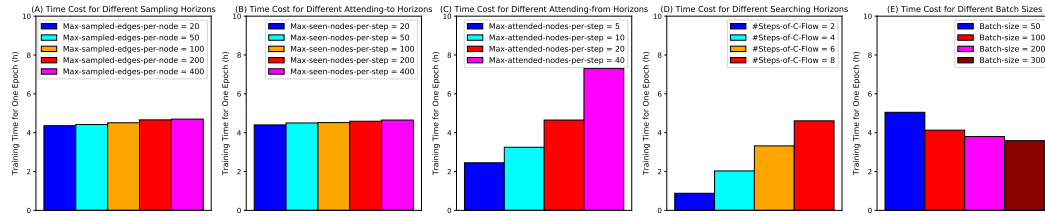


Figure 6: Analysis of time cost on WN18RR: (A)-(D) measure the training time for one epoch on different horizon settings corresponding to Figure 4(C)-(F); (E) measures the training time for one epoch for different batch sizes using the same horizon setting, which is $Max\text{-sampled-edges-per-node}=20$, $Max\text{-seen-nodes-per-step}=20$, $Max\text{-attended-nodes-per-step}=20$, and $\#Steps\text{-of-C-Flow}=8$. The time cost analysis charts on FB15K-237 can be found in the appendix.

4.4 Visualization

To further demonstrate the reasoning ability acquired by our model, we show some visualization results of the extracted subgraphs on NELL995's test data for 12 separate tasks. We avoid using the training data in order to show the generalization of our model's learned reasoning ability on knowledge graphs. Here, we show the visualization result for the *AthletePlaysForTeam* task. The rest can be found in the appendix.

For the AthletePlaysForTeam task

Query: (concept_personnorthamerica_michael_turner, concept_athleteplaysforteam, concept_sportsteam_falcons)

Selected key edges:

concept_personnorthamerica_michael_turner, concept_agentbelongstoorganization, concept_sportsleague_nfl
concept_personnorthamerica_michael_turner, concept_athlethomestadium, concept_stadiumoreventvenue_georgia_dome
concept_sportsleague_nfl, concept_agentcompeteswithagent, concept_sportsleague_nfl
concept_sportsleague_nfl, concept_agentcompeteswithagent_inv, concept_sportsleague_nfl
concept_sportsleague_nfl, concept_teamplaysinleague_inv, concept_sportsteam_sd_chargers
concept_sportsleague_nfl, concept_leaguestadiums, concept_stadiumoreventvenue_georgia_dome
concept_sportsleague_nfl, concept_teamplaysinleague_inv, concept_sportsteam_falcons
concept_sportsleague_nfl, concept_agentbelongstoorganization_inv, concept_personnorthamerica_michael_turner
concept_stadiumoreventvenue_georgia_dome, concept_leaguestadiums_inv, concept_sportsleague_nfl
concept_stadiumoreventvenue_georgia_dome, concept_teamhomestadium_inv, concept_sportsteam_falcons
concept_stadiumoreventvenue_georgia_dome, concept_athlethomestadium_inv, concept_athlete_joey_harrington
concept_stadiumoreventvenue_georgia_dome, concept_athlethomestadium_inv, concept_athlete_roddey_white
concept_stadiumoreventvenue_georgia_dome, concept_athlethomestadium_inv, concept_coach_deangelo_hall
concept_stadiumoreventvenue_georgia_dome, concept_athlethomestadium_inv, concept_personnorthamerica_michael_turner
concept_sportsleague_nfl, concept_subpartoforganization_inv, concept_sportsteam_oakland_raiders
concept_sportsteam_sd_chargers, concept_teamplaysinleague, concept_sportsleague_nfl
concept_sportsteam_sd_chargers, concept_teamplaysagainstteam, concept_sportsteam_falcons
concept_sportsteam_sd_chargers, concept_teamplaysagainstteam_inv, concept_sportsteam_falcons
concept_sportsteam_sd_chargers, concept_teamplaysagainstteam, concept_sportsteam_oakland_raiders
concept_sportsteam_sd_chargers, concept_teamplaysagainstteam_inv, concept_sportsteam_oakland_raiders
concept_sportsteam_falcons, concept_teamplaysinleague, concept_sportsleague_nfl
concept_sportsteam_falcons, concept_teamplaysagainstteam, concept_sportsteam_sd_chargers
concept_sportsteam_falcons, concept_teamplaysagainstteam_inv, concept_sportsteam_sd_chargers
concept_sportsteam_falcons, concept_teamhomestadium, concept_stadiumoreventvenue_georgia_dome

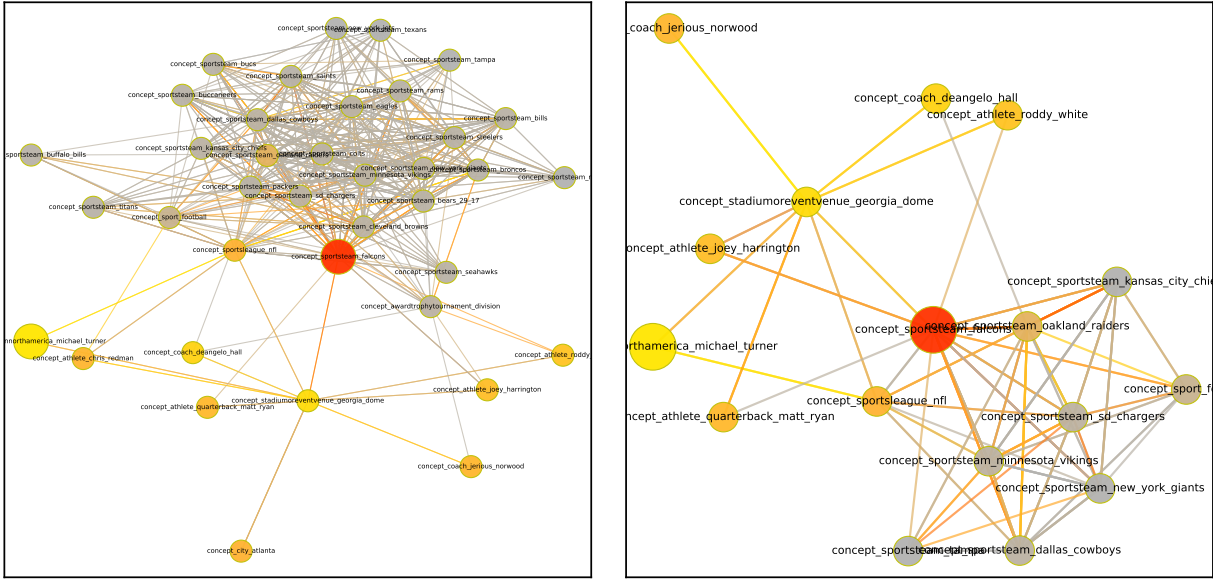


Figure 7: **AthletePlaysForTeam**. The head is `concept_personnorthamerica_michael_turner`, the query relation is `concept:athleteplaysforteam`, and the desired tail is `concept_sportsteam_falcons`. The left is a full subgraph derived with `max_attended_nodes_per_step = 20`, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

```

concept_sportsteam_falcons , concept:teamplaysagainstteam , concept_sportsteam_oakland_raiders
concept_sportsteam_falcons , concept:teamplaysagainstteam_inv , concept_sportsteam_oakland_raiders
concept_sportsteam_falcons , concept:athleleedsportsteam_inv , concept_athlete_joey_harrington
concept_athlete_joey_harrington , concept:athlethomestadium , concept_stadiumoreventvenue_georgia_dome
concept_athlete_joey_harrington , concept:athleleedsportsteam , concept_sportsteam_falcons
concept_athlete_joey_harrington , concept:athleteplaysforteam , concept_sportsteam_falcons
concept_athlete_rodny_white , concept:athlethomestadium , concept_stadiumoreventvenue_georgia_dome
concept_athlete_rodny_white , concept:athleteplaysforteam , concept_sportsteam_falcons
concept_coach_deangelo_hall , concept:athlethomestadium , concept_stadiumoreventvenue_georgia_dome
concept_coach_deangelo_hall , concept:athleteplaysforteam , concept_sportsteam_oakland_raiders
concept_sportsleague_nfl , concept:teamplaysinleague_inv , concept_sportsteam_new_york_giants
concept_sportsteam_sd_chargers , concept:teamplaysagainstteam_inv , concept_sportsteam_new_york_giants
concept_sportsteam_falcons , concept:teamplaysagainstteam , concept_sportsteam_new_york_giants
concept_sportsteam_falcons , concept:teamplaysagainstteam_inv , concept_sportsteam_new_york_giants
concept_sportsteam_oakland_raiders , concept:teamplaysagainstteam_inv , concept_sportsteam_new_york_giants
concept_sportsteam_oakland_raiders , concept:teamplaysagainstteam , concept_sportsteam_sd_chargers
concept_sportsteam_oakland_raiders , concept:teamplaysagainstteam_inv , concept_sportsteam_sd_chargers
concept_sportsteam_oakland_raiders , concept:teamplaysagainstteam , concept_sportsteam_falcons
concept_sportsteam_oakland_raiders , concept:teamplaysagainstteam_inv , concept_sportsteam_falcons
concept_sportsteam_oakland_raiders , concept:agentcompeteswithagent , concept_sportsteam_oakland_raiders
concept_sportsteam_oakland_raiders , concept:agentcompeteswithagent_inv , concept_sportsteam_oakland_raiders
concept_sportsteam_new_york_giants , concept:teamplaysagainstteam , concept_sportsteam_sd_chargers
concept_sportsteam_new_york_giants , concept:teamplaysagainstteam , concept_sportsteam_falcons
concept_sportsteam_new_york_giants , concept:teamplaysagainstteam_inv , concept_sportsteam_falcons
concept_sportsteam_new_york_giants , concept:teamplaysagainstteam , concept_sportsteam_oakland_raiders

```

In the above case, the query is $(concept_personnorthamerica_michael_turner, concept:athleteplaysforteam, ?)$ and the desired answer is `concept_sportsteam_falcons`. From Figure 7, we can see our model learns that $(concept_personnorthamerica_michael_turner, concept:athlethomestadium, concept_stadiumoreventvenue_georgia_dome)$ and $(concept_stadiumoreventvenue_georgia_dome, concept:teamhomestadium_inv, concept_sportsteam_falcons)$ are two important facts to support the answer of `concept_sportsteam_falcons`. Besides, other facts, such as $(concept_athlete_joey_harrington, concept:athlethomestadium, concept_stadiumoreventvenue_georgia_dome)$ and $(concept_athlete_joey_harrington, concept:athleteplaysforteam, concept_sportsteam_falcons)$, provide a vivid example that a person or an athlete with `concept_stadiumoreventvenue_georgia_dome` as his or her home stadium might play for the team `concept_sportsteam_falcons`. We have such examples more than one, like `concept_athlete_rodny_white`'s and `concept_athlete_quarterback_matt_ryan`'s. The entity

`concept_sportsleague_nfl` cannot help us differentiate the true answer from other NFL teams, but it can at least exclude those non-NFL teams. In a word, our subgraph-structured representation can well capture the relational and compositional reasoning pattern.

5 Conclusion

We introduce an attentive message passing mechanism on graphs under the notion of attentive awareness, inspired by the phenomenon of consciousness, to model the iterative compositional reasoning pattern by forming a compact query-dependent subgraph. We propose an attentive computation framework with three flow-based layer to combine GNNs' representation power with explicit reasoning process, and further reduce the complexity when applying GNNs to large-scale graphs. It is worth mentioning that our framework is not limited to knowledge graph reasoning, but has a wider applicability to large-scale graph-based computation with a few input-dependent nodes and edges involved each time.

References

- [1] Yoshua Bengio. The consciousness prior. *CoRR*, abs/1709.08568, 2017.
- [2] Stanislas Dehaene, Michel Kerszberg, and Jean Pierre Changeux. A neuronal model of a global workspace in effortful cognitive tasks. *Proceedings of the National Academy of Sciences of the United States of America*, 95 24:14529–34, 1998.
- [3] Giulio Tononi, Mélanie Boly, Marcello Massimini, and Christof Koch. Integrated information theory: from consciousness to its physical substrate. *Nature Reviews Neuroscience*, 17:450–461, 2016.
- [4] David Rosenthal and Josh Weisberg. Higher-order theories of consciousness. *Scholarpedia*, 3:4407, 2008.
- [5] Robert Van Gulick. Higher-order global states (hogs): an alternative higher-order model. *Higher-order theories of consciousness*, pages 67–93, 2004.
- [6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.
- [7] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çağlar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.
- [8] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.
- [9] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2015.
- [10] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.
- [11] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, 2016.
- [12] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *CoRR*, abs/1902.10197, 2018.
- [13] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, 2018.

- [14] Ni Lao, Tom Michael Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.
- [15] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, 2017.
- [16] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alexander J. Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *CoRR*, abs/1711.05851, 2018.
- [17] Yelong Shen, Jianshu Chen, Pu Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using monte carlo tree search. In *NeurIPS*, 2018.
- [18] William W. Cohen. Tensorlog: A differentiable deductive database. *CoRR*, abs/1605.06523, 2016.
- [19] Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, 2017.
- [20] Xiaoran Xu, Songpeng Zu, Chengliang Gao, Yuan Zhang, and Wei Feng. Modeling attention flow on graphs. *CoRR*, abs/1811.00497, 2018.
- [21] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.
- [22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.
- [23] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jian Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, 2015.
- [24] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Michael Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *EMNLP*, 2014.
- [25] Kelvin Guu, John Miller, and Percy S. Liang. Traversing knowledge graphs in vector space. In *EMNLP*, 2015.
- [26] Yankai Lin, Zhiyuan Liu, and Maosong Sun. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, 2015.
- [27] Kristina Toutanova, Victoria Lin, Wen tau Yih, Hoifung Poon, and Chris Quirk. Compositional learning of embeddings for relation paths in knowledge base and text. In *ACL*, 2016.
- [28] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*, 2017.
- [29] Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*, 2016.
- [30] Kenneth H. Craik. The nature of explanation. 1952.
- [31] John R. Anderson. Acquisition of cognitive skill. 1982.
- [32] Dedre Gentner and Arthur B. Markman. Structure mapping in analogy and similarity. 1997.
- [33] John E. Hummel and Keith J. Holyoak. A symbolic-connectionist theory of relational inference and generalization. *Psychological review*, 110 2:220–64, 2003.
- [34] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *The Behavioral and brain sciences*, 40:e253, 2017.
- [35] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2014.

- [36] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.
- [37] David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.
- [38] Steven M. Kearnes, Kevin McCloskey, Marc Berndl, Vijay S. Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30 8:595–608, 2016.
- [39] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.
- [40] Mathias Niepert, Mohammed Hassan Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.
- [41] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2017.
- [42] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34:18–42, 2017.
- [43] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. *CoRR*, abs/1511.05493, 2016.
- [44] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.
- [45] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NIPS*, 2016.
- [46] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.
- [47] Michael Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *CoRR*, abs/1612.00341, 2017.
- [48] Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *ICML*, 2018.
- [49] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin A. Riedmiller, Raia Hadsell, and Peter W. Battaglia. Graph networks as learnable physics engines for inference and control. In *ICML*, 2018.
- [50] Jessica B. Hamrick, Kelsey R. Allen, Victor Bapst, Tina Zhu, Kevin R. McKee, Joshua B. Tenenbaum, and Peter W. Battaglia. Relational inductive bias for physical construction in humans and machines. *CoRR*, abs/1806.01203, 2018.
- [51] Nicholas Watters, Daniel Zoran, Théophane Weber, Peter W. Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *NIPS*, 2017.
- [52] David Raposo, Adam Santoro, David G. T. Barrett, Razvan Pascanu, Timothy P. Lillicrap, and Peter W. Battaglia. Discovering objects and their relations from entangled scene representations. *CoRR*, abs/1702.05068, 2017.
- [53] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

- [54] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018.
- [55] Adam Santoro, Ryan Faulkner, David Raposo, Jack W. Rae, Mike Chrzanowski, Théophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy P. Lillicrap. Relational recurrent neural networks. In *NeurIPS*, 2018.
- [56] Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *NeurIPS*, 2018.
- [57] Daniel Oñoro-Rubio, Mathias Niepert, Alberto García-Durán, Roberto Gonzalez, and Roberto Javier López-Sastre. Representation learning for visual-relational knowledge graphs. *CoRR*, abs/1709.02314, 2017.
- [58] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. 2017.
- [59] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *NIPS*, 2016.
- [60] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. In *NIPS*, 2017.
- [61] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. *CoRR*, abs/1711.00740, 2018.
- [62] Irwan Bello, Hieu Quang Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940, 2017.
- [63] Alex Nowak, Soledad Villar, Afonso S. Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *CoRR*, abs/1706.07450, 2017.
- [64] Elias Boutros Khalil, Hanjun Dai, Yuyu Zhang, Bistra N. Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, 2017.
- [65] Daniel D. Johnson. Learning graphical state transitions. In *ICLR*, 2017.
- [66] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy S. Liang, Leonardo de Moura, and David L. Dill. Learning a sat solver from single-bit supervision. *CoRR*, abs/1802.03685, 2018.
- [67] Jessica B. Hamrick, Andrew J. Ballard, Razvan Pascanu, Oriol Vinyals, Nicolas Heess, and Peter W. Battaglia. Metacontrol for adaptive imagination-based optimization. *CoRR*, abs/1705.02670, 2017.
- [68] Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sébastien Racanière, David P. Reichert, Théophane Weber, Daan Wierstra, and Peter W. Battaglia. Learning model-based planning from scratch. *CoRR*, abs/1707.06170, 2017.
- [69] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.
- [70] Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter W. Battaglia. Relational deep reinforcement learning. *CoRR*, abs/1806.01830, 2018.
- [71] Sam Toyer, Felipe W. Trevizan, Sylvie Thiébaux, and Lexing Xie. Action schema networks: Generalised policies with deep learning. In *AAAI*, 2018.
- [72] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.
- [73] Risi Kondor, Hy Truong Son, Horace Pan, Brandon M. Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *CoRR*, abs/1801.02144, 2018.

- [74] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Alejandro Romero, Pietro Lió, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2018.
- [75] Wouter Kool. Attention solves your tsp , approximately. 2018.
- [76] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.
- [77] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017.
- [78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- [79] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. Learning deep generative models of graphs. *CoRR*, abs/1803.03324, 2018.
- [80] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *CoRR*, abs/1805.11973, 2018.
- [81] Jiaxuan You, Zhitao Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, 2018.
- [82] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *ICML*, 2018.
- [83] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT*, 2018.
- [84] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. 2015.
- [85] Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2014.
- [86] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, 2016.

6 Appendix

6.1 Hyperparameter settings

Table 6: The standard hyperparameter settings we use for each dataset plus their training time for one epoch. For the experimental analysis, we only adjust one hyperparameter and keep the remaining fixed at the standard setting. For NELL995, the training time per epoch means the average time cost of the 12 single-query-relation tasks.

Hyperparameter	FB15K-237	FB15K	WN18RR	WN18	YAGO3-10	NELL995
<i>batch_size</i>	80	80	100	100	100	10
<i>n_dims_att</i>	50	50	50	50	50	200
<i>n_dims</i>	100	100	100	100	100	200
<i>max_sampled_edges_per_step</i>	10000	10000	10000	10000	10000	10000
<i>max_attended_nodes_per_step</i>	20	20	20	20	20	100
<i>max_sampled_edges_per_node</i>	200	200	200	200	200	1000
<i>max_seen_nodes_per_step</i>	200	200	200	200	200	1000
<i>n_steps_of_u_flow</i>	2	1	2	1	1	1
<i>n_steps_of_c_flow</i>	6	6	8	8	6	5
<i>learning_rate</i>	0.001	0.001	0.001	0.001	0.0001	0.001
<i>optimizer</i>	Adam	Adam	Adam	Adam	Adam	Adam
<i>grad_clipnorm</i>	1	1	1	1	1	1
<i>n_epochs</i>	1	1	1	1	1	3
Training time per epoch (h)	25.7	63.7	4.3	8.5	185.0	0.12

Our hyperparameters can be categorized into three groups:

- The normal hyperparameters, including *batch_size*, *n_dims_att*, *n_dims*, *learning_rate*, *grad_clipnorm*, and *n_epochs*. Here, we set a smaller dimension, *n_dims_att*, for the attention flow computation, as it uses more edges for computation than the message passing uses in the consciousness flow layer, and also intuitively, it does not need to propagate high-dimensional messages but only compute a scalar score for each of the sampled neighbor nodes, in concert with the idea in the key-value mechanism [1]. We set *n_epochs* = 1 in most cases, indicating that our model needs to be trained only for one epoch due to its fast convergence.
- The hyperparameters that are in charge of controlling the sampling-attending horizon, including *max_sampled_edges_per_step* that controls the maximum number to sample edges per step per query for the message passing in the unconsciousness flow layer, and *max_sampled_edges_per_node*, *max_attended_nodes_per_step* and *max_seen_nodes_per_step* that control the maximum number to sample edges connected to each current node per step per query, the maximum number of current nodes to attend from per step per query, and the maximum number of neighbor nodes to attend to per step per query in the consciousness flow layer.
- The hyperparameters that are in charge of controlling the searching horizon, including *n_steps_of_u_flow* representing the number of steps to run the unconsciousness flow, and *n_steps_of_c_flow* representing the number of steps to run the consciousness flow.

Note that we choose these hyperparameters not only by their performances but also the computation resources available to us. In some cases, to deal with a very large knowledge graph with limited resources, we need to make a trade-off between the efficiency and the effectiveness. For example, each of NELL995’s single-query-relation tasks has a small training set, though still with a large graph, so we can reduce the batch size in favor of affording larger dimensions and a larger sampling-attending horizon without any concern for waiting too long to finish one epoch.

6.2 Other experimental analysis

See Figure 8,9.

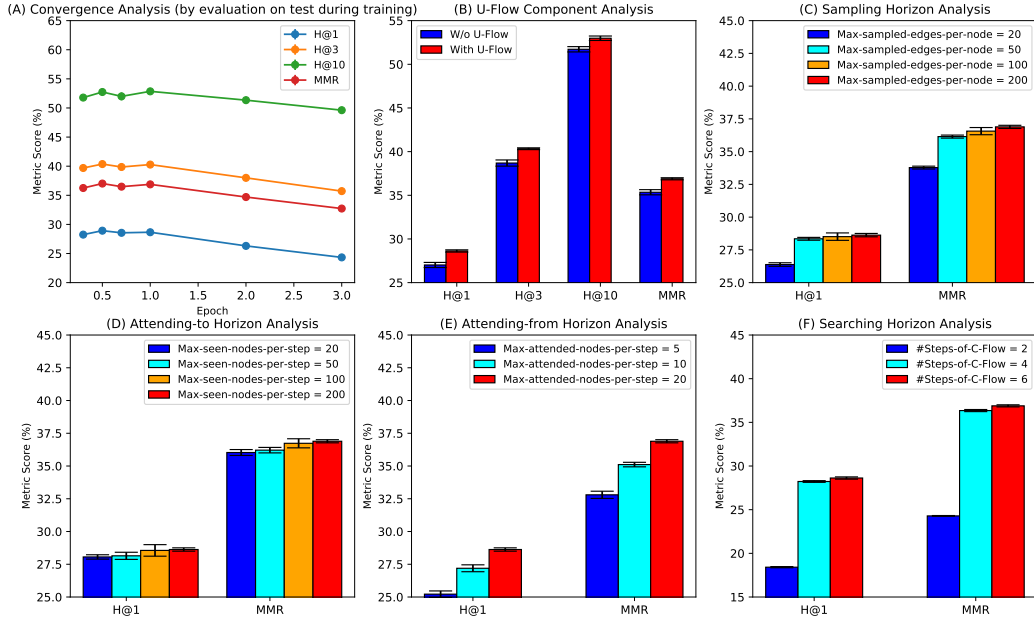


Figure 8: Experimental analysis on FB15K-237: (A) During training we pick six model snapshots at time points of 0.3, 0.5, 0.7, 1, 2, and 3 epochs and evaluate them on test; (B) The *w/o U-Flow* uses zero step to run U-Flow, while the *with U-Flow* uses two steps; (C)-(F) are for the sampling, attending and searching horizon analysis based on the standard hyperparameter settings listed in the appendix.

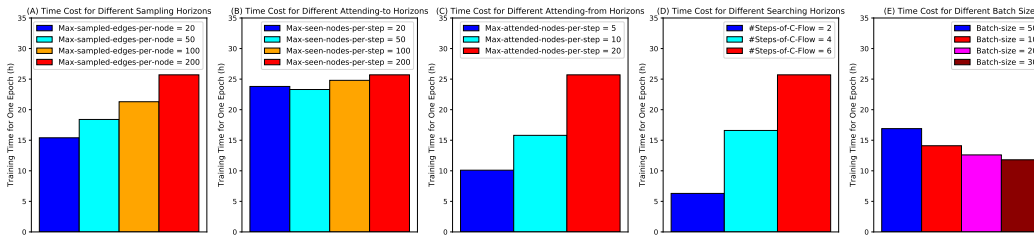


Figure 9: Analysis of time cost on FB15K-237: (A)-(D) measure the training time for one epoch on different horizon settings corresponding to Figure 8(C)-(F); (E) measures the training time for one epoch for different batch sizes using the same horizon setting, which is *Max-sampled-edges-per-node=20*, *Max-seen-nodes-per-step=20*, *Max-attended-nodes-per-step=20*, and *#Steps-of-C-Flow=6*.

6.3 Other visualization

For the AthletePlaysInLeague task

Query: (concept_personnorthamerica_matt_treanor, concept:athleteplaysinleague, concept_sportsleague_mlb)

Selected key edges:

concept_personnorthamerica_matt_treanor, concept:athleteflyouttosportsteamposition, concept_sportsteamposition_center
concept_personnorthamerica_matt_treanor, concept:athletelaysport, concept_sport_baseball
concept_sportsteamposition_center, concept:athleteflyouttosportsteamposition_inv, concept_personus_orlando_hudson
concept_sportsteamposition_center, concept:athleteflyouttosportsteamposition_inv, concept_athlete_ben_hendrickson
concept_sportsteamposition_center, concept:athleteflyouttosportsteamposition_inv, concept_coach_j_j_hardy
concept_sportsteamposition_center, concept:athleteflyouttosportsteamposition_inv, concept_athlete_hunter_pence
concept_sport_baseball, concept:athletelaysport_inv, concept_personus_orlando_hudson
concept_sport_baseball, concept:athletelaysport_inv, concept_athlete_ben_hendrickson
concept_sport_baseball, concept:athletelaysport_inv, concept_coach_j_j_hardy
concept_sport_baseball, concept:athletelaysport_inv, concept_athlete_hunter_pence
concept_personus_orlando_hudson, concept:athletelaysinleague, concept_sportsleague_mlb
concept_personus_orlando_hudson, concept:athletelaysport, concept_sport_baseball
concept_athlete_ben_hendrickson, concept:coachesinleague, concept_sportsleague_mlb
concept_athlete_ben_hendrickson, concept:athletelaysport, concept_sport_baseball

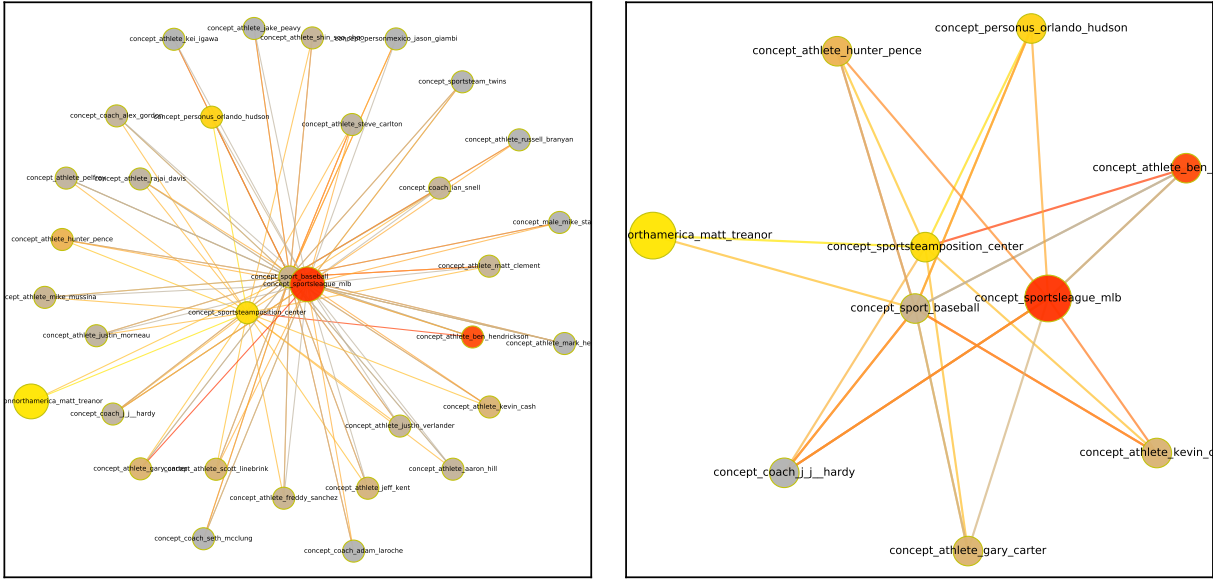


Figure 10: **AthletePlaysInLeague**. The head is `concept:athleteplaysinleague`, the query relation is `concept:athleteplaysinleague`, and the desired tail is `.`. The left is a full subgraph derived with $max_attended_nodes_per_step = 20$, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

`concept_coach_j_j_hardy`, `concept:coachesinleague`, `concept_sportsleague_mlb`
`concept_coach_j_j_hardy`, `concept:athleteplaysinleague`, `concept_sportsleague_mlb`
`concept_coach_j_j_hardy`, `concept:athleteplayssport`, `concept_sport_baseball`
`concept_athlete_hunter_pence`, `concept:athleteplaysinleague`, `concept_sportsleague_mlb`
`concept_athlete_hunter_pence`, `concept:athleteplayssport`, `concept_sport_baseball`
`concept_sportsleague_mlb`, `concept:coachesinleague_inv`, `concept_athlete_ben_hendrickson`
`concept_sportsleague_mlb`, `concept:coachesinleague_inv`, `concept_coach_j_j_hardy`

For the AthleteHomeStadium task

Query: (`concept_athlete_eli_manning`, `concept:athlethomestadium`, `concept_stadiumeventvenue_giants_stadium`)

Selected key edges:

`concept_athlete_eli_manning`, `concept:personbelongstoorganization`, `concept_sportsteam_new_york_giants`
`concept_athlete_eli_manning`, `concept:athleteplaysforteam`, `concept_sportsteam_new_york_giants`
`concept_athlete_eli_manning`, `concept:athleleedsportsteam`, `concept_sportsteam_new_york_giants`
`concept_athlete_eli_manning`, `concept:athleteplaysinleague`, `concept_sportsleague_nfl`
`concept_athlete_eli_manning`, `concept:fatherofperson_inv`, `concept_male_archie_manning`
`concept_sportsteam_new_york_giants`, `concept:teamplaysinleague`, `concept_sportsleague_nfl`
`concept_sportsteam_new_york_giants`, `concept:teamhomestadium`, `concept_stadiumeventvenue_giants_stadium`
`concept_sportsteam_new_york_giants`, `concept:personbelongstoorganization_inv`, `concept_athlete_eli_manning`
`concept_sportsteam_new_york_giants`, `concept:athleteplaysforteam_inv`, `concept_athlete_eli_manning`
`concept_sportsteam_new_york_giants`, `concept:athleleedsportsteam_inv`, `concept_athlete_eli_manning`
`concept_sportsleague_nfl`, `concept:teamplaysinleague_inv`, `concept_sportsteam_new_york_giants`
`concept_sportsleague_nfl`, `concept:agentcompeteswithagent`, `concept_sportsleague_nfl`
`concept_sportsleague_nfl`, `concept:agentcompeteswithagent_inv`, `concept_sportsleague_nfl`
`concept_sportsleague_nfl`, `concept:leaguestadiums`, `concept_stadiumeventvenue_giants_stadium`
`concept_sportsleague_nfl`, `concept:athleteplaysinleague_inv`, `concept_athlete_eli_manning`
`concept_male_archie_manning`, `concept:fatherofperson`, `concept_athlete_eli_manning`
`concept_sportsleague_nfl`, `concept:leaguestadiums`, `concept_stadiumeventvenue_paul_brown_stadium`
`concept_stadiumeventvenue_giants_stadium`, `concept:teamhomestadium_inv`, `concept_sportsteam_new_york_giants`
`concept_stadiumeventvenue_giants_stadium`, `concept:leaguestadiums_inv`, `concept_sportsleague_nfl`
`concept_stadiumeventvenue_giants_stadium`, `concept:proxyfor_inv`, `concept_city_east_rutherford`
`concept_city_east_rutherford`, `concept:proxyfor`, `concept_stadiumeventvenue_giants_stadium`
`concept_stadiumeventvenue_paul_brown_stadium`, `concept:leaguestadiums_inv`, `concept_sportsleague_nfl`

For the AthletePlaysSport task

Query: (`concept_athlete_vernon_wells`, `concept:athleteplayssport`, `concept_sport_baseball`)

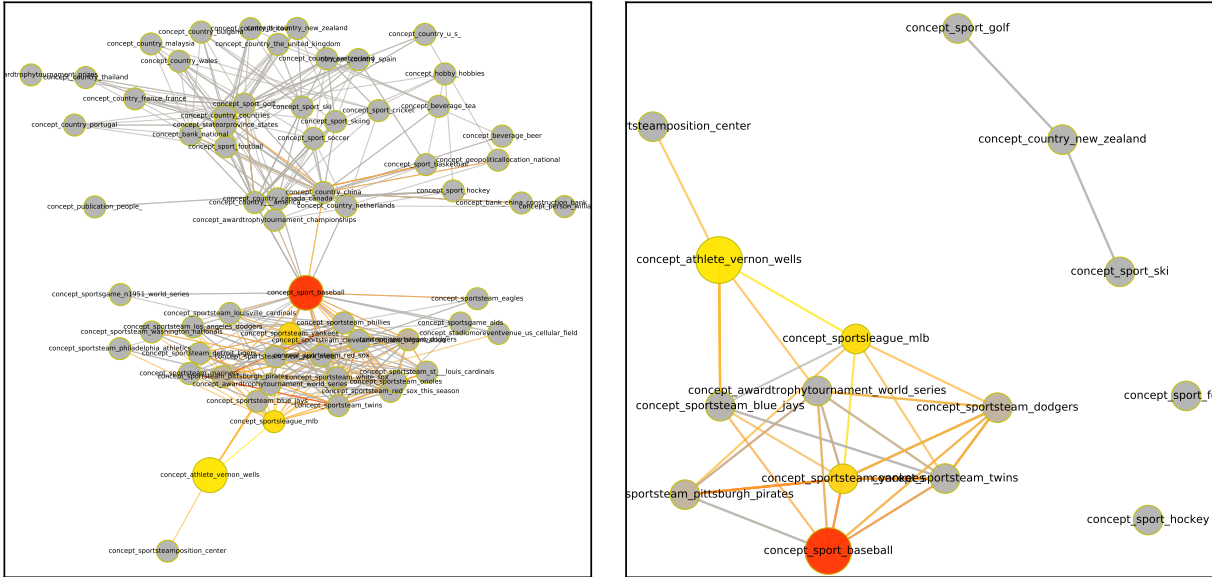


Figure 12: **AthletePlaysSport**. The head is *concept_athlete_vernon_wells*, the query relation is *concept:athleteplaysport*, and the desired tail is *concept_sport_baseball*. The left is a full subgraph derived with *max_attended_nodes_per_step = 20*, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

For the TeamPlaysSport task

Query: (*concept_sportsteam_red_wings* , *concept:teamplaysport* , *concept_sport_hockey*)

Selected key edges:

concept_sportsteam_red_wings , *concept:teamplaysagainstteam* , *concept_sportsteam_montreal_canadiens*
concept_sportsteam_red_wings , *concept:teamplaysagainstteam_inv* , *concept_sportsteam_montreal_canadiens*
concept_sportsteam_red_wings , *concept:teamplaysagainstteam* , *concept_sportsteam_blue_jackets*
concept_sportsteam_red_wings , *concept:teamplaysagainstteam_inv* , *concept_sportsteam_blue_jackets*
concept_sportsteam_red_wings , *concept:worksfor_inv* , *concept_athlete_lidstrom*
concept_sportsteam_red_wings , *concept:organizationhireperson* , *concept_athlete_lidstrom*
concept_sportsteam_red_wings , *concept:athleteplaysforteam_inv* , *concept_athlete_lidstrom*
concept_sportsteam_red_wings , *concept:athleleledsportsteam_inv* , *concept_athlete_lidstrom*
concept_sportsteam_montreal_canadiens , *concept:teamplaysagainstteam* , *concept_sportsteam_red_wings*
concept_sportsteam_montreal_canadiens , *concept:teamplaysagainstteam_inv* , *concept_sportsteam_red_wings*
concept_sportsteam_montreal_canadiens , *concept:teamplaysinleague* , *concept_sportsleague_nhl*
concept_sportsteam_montreal_canadiens , *concept:teamplaysagainstteam* , *concept_sportsteam_leafs*
concept_sportsteam_montreal_canadiens , *concept:teamplaysagainstteam_inv* , *concept_sportsteam_leafs*
concept_sportsteam_blue_jackets , *concept:teamplaysagainstteam* , *concept_sportsteam_red_wings*
concept_sportsteam_blue_jackets , *concept:teamplaysagainstteam_inv* , *concept_sportsteam_red_wings*
concept_sportsteam_blue_jackets , *concept:teamplaysinleague* , *concept_sportsleague_nhl*
concept_athlete_lidstrom , *concept:worksfor* , *concept_sportsteam_red_wings*
concept_athlete_lidstrom , *concept:organizationhireperson_inv* , *concept_sportsteam_red_wings*
concept_athlete_lidstrom , *concept:athleteplaysforteam* , *concept_sportsteam_red_wings*
concept_athlete_lidstrom , *concept:athleleledsportsteam* , *concept_sportsteam_red_wings*
concept_sportsteam_red_wings , *concept:teamplaysinleague* , *concept_sportsleague_nhl*
concept_sportsteam_red_wings , *concept:teamplaysagainstteam* , *concept_sportsteam_leafs*
concept_sportsteam_red_wings , *concept:teamplaysagainstteam_inv* , *concept_sportsteam_leafs*
concept_sportsleague_nhl , *concept:agentcompeteswithagent* , *concept_sportsleague_nhl*
concept_sportsleague_nhl , *concept:agentcompeteswithagent_inv* , *concept_sportsleague_nhl*
concept_sportsleague_nhl , *concept:teamplaysinleague_inv* , *concept_sportsteam_leafs*
concept_sportsteam_leafs , *concept:teamplaysinleague* , *concept_sportsleague_nhl*
concept_sportsteam_leafs , *concept:teamplaysport* , *concept_sport_hockey*

For the OrganizationHeadQuarteredInCity task

Query: (*concept_company_disney* , *concept:organizationheadquarteredincity* , *concept_city_burbank*)

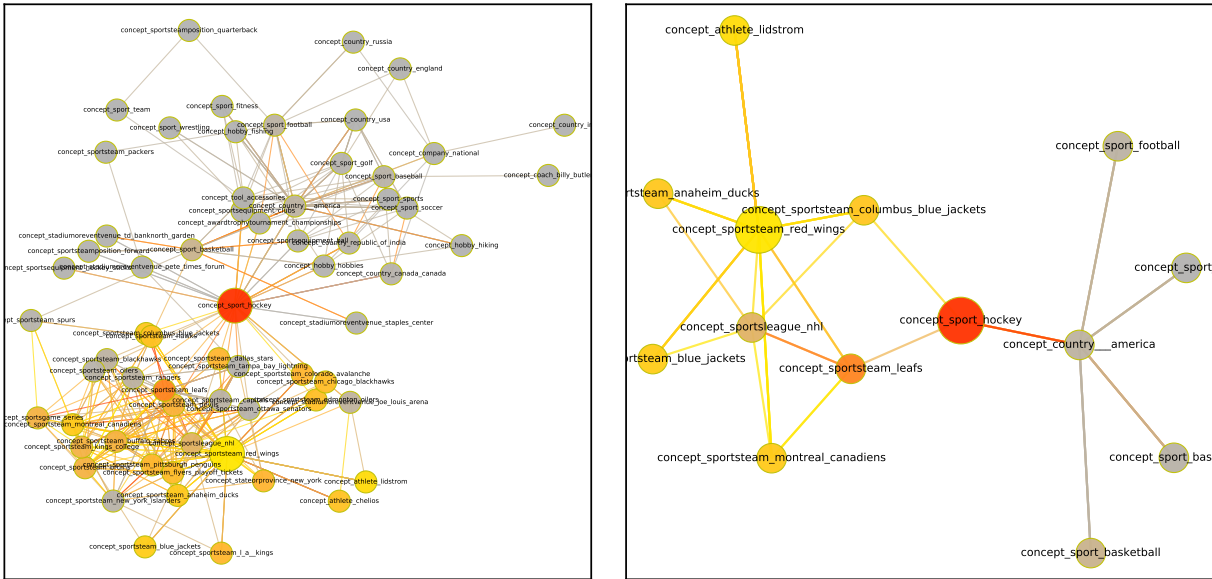


Figure 13: **TeamPlaysSport**. The head is *concept_sportsteam_red_wings*, the query relation is *concept:teamplaysport*, and the desired tail is *concept_sport_hockey*. The left is a full subgraph derived with *max_attended_nodes_per_step = 20*, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

Selected key edges:

concept_company_disney, concept:headquartered_in, concept_city_burbank
 concept_company_disney, concept:subpartoforganization_inv, concept_website_network
 concept_company_disney, concept:worksfor_inv, concept_ceo_robert_iger
 concept_company_disney, concept:proxyfor_inv, concept_ceo_robert_iger
 concept_company_disney, concept:personleadsorganization_inv, concept_ceo_robert_iger
 concept_company_disney, concept:ceof_inv, concept_ceo_robert_iger
 concept_company_disney, concept:personleadsorganization_inv, concept_ceo_jeffrey_katzenberg
 concept_company_disney, concept:organizationhireperson, concept_ceo_jeffrey_katzenberg
 concept_company_disney, concept:organizationterminatedperson, concept_ceo_jeffrey_katzenberg
 concept_city_burbank, concept:headquartered_in_inv, concept_company_disney
 concept_city_burbank, concept:headquartered_in_inv, concept_biotechcompany_the_walt_disney_co_
 concept_website_network, concept:subpartoforganization, concept_company_disney
 concept_ceo_robert_iger, concept:worksfor, concept_company_disney
 concept_ceo_robert_iger, concept:proxyfor, concept_company_disney
 concept_ceo_robert_iger, concept:personleadsorganization, concept_company_disney
 concept_ceo_robert_iger, concept:ceof, concept_company_disney
 concept_ceo_robert_iger, concept:topmemberoforganization, concept_biotechcompany_the_walt_disney_co_
 concept_ceo_robert_iger, concept:organizationterminatedperson_inv, concept_biotechcompany_the_walt_disney_co_
 concept_ceo_jeffrey_katzenberg, concept:personleadsorganization, concept_company_disney
 concept_ceo_jeffrey_katzenberg, concept:organizationhireperson_inv, concept_company_disney
 concept_ceo_jeffrey_katzenberg, concept:organizationterminatedperson_inv, concept_company_disney
 concept_ceo_jeffrey_katzenberg, concept:worksfor, concept_recordlabel_dreamworks_skg
 concept_ceo_jeffrey_katzenberg, concept:topmemberoforganization, concept_recordlabel_dreamworks_skg
 concept_ceo_jeffrey_katzenberg, concept:organizationterminatedperson_inv, concept_recordlabel_dreamworks_skg
 concept_ceo_jeffrey_katzenberg, concept:ceof, concept_recordlabel_dreamworks_skg
 concept_biotechcompany_the_walt_disney_co_, concept:headquartered_in, concept_city_burbank
 concept_biotechcompany_the_walt_disney_co_, concept:organizationheadquartered_in_city, concept_city_burbank
 concept_recordlabel_dreamworks_skg, concept:worksfor_inv, concept_ceo_jeffrey_katzenberg
 concept_recordlabel_dreamworks_skg, concept:topmemberoforganization_inv, concept_ceo_jeffrey_katzenberg
 concept_recordlabel_dreamworks_skg, concept:organizationterminatedperson, concept_ceo_jeffrey_katzenberg
 concept_recordlabel_dreamworks_skg, concept:ceof_inv, concept_ceo_jeffrey_katzenberg
 concept_city_burbank, concept:airportincity_inv, concept_transportation_burbank_glendale_pasadena
 concept_transportation_burbank_glendale_pasadena, concept:airportincity, concept_city_burbank

For the WorksFor task

Query: (concept_scientist_balmer, concept:worksfor, concept_university_microsoft)

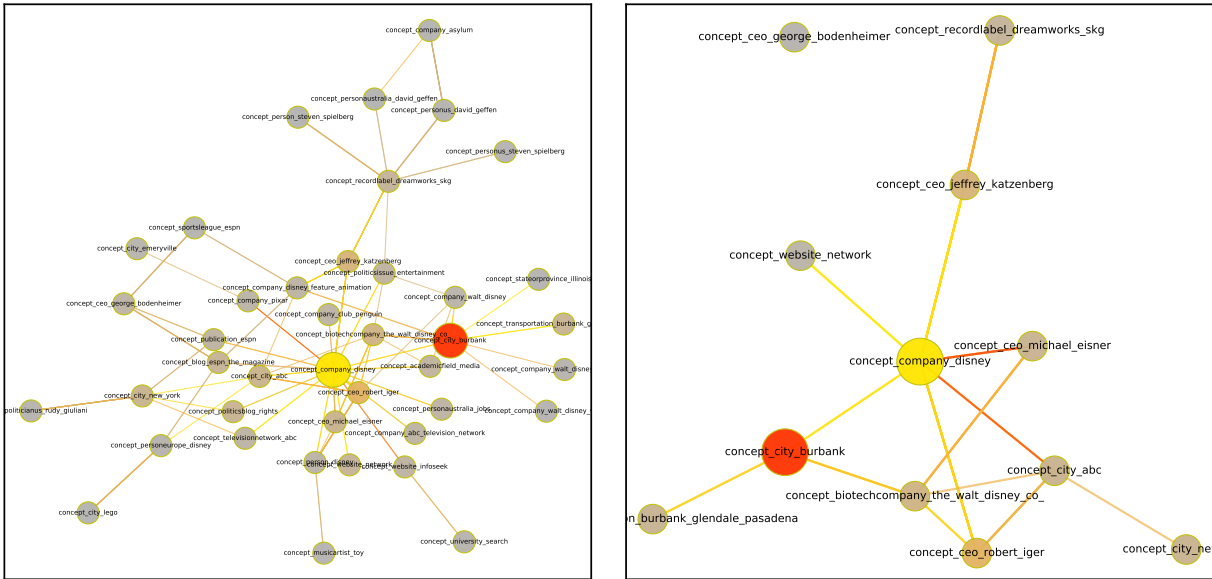


Figure 14: **OrganizationHeadQuarteredInCity**. The head is `concept_company_disney`, the query relation is `concept:organizationheadquarteredincity`, and the desired tail is `concept_city_burbank`. The left is a full subgraph derived with `max_attended_nodes_per_step = 20`, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

Selected key edges:

`concept_scientist_balmer`, `concept:topmemberoforganization`, `concept_company_microsoft`
`concept_scientist_balmer`, `concept:organizationterminatedperson_inv`, `concept_university_microsoft`
`concept_company_microsoft`, `concept:topmemberoforganization_inv`, `concept_personus_steve_ballmer`
`concept_company_microsoft`, `concept:topmemberoforganization_inv`, `concept_scientist_balmer`
`concept_university_microsoft`, `concept:agentcollaborateswithagent`, `concept_personus_steve_ballmer`
`concept_university_microsoft`, `concept:personleadsorganization_inv`, `concept_personus_steve_ballmer`
`concept_university_microsoft`, `concept:personleadsorganization_inv`, `concept_person_bill`
`concept_university_microsoft`, `concept:organizationterminatedperson`, `concept_scientist_balmer`
`concept_university_microsoft`, `concept:personleadsorganization_inv`, `concept_person_robbie_bach`
`concept_personus_steve_ballmer`, `concept:topmemberoforganization`, `concept_company_microsoft`
`concept_personus_steve_ballmer`, `concept:agentcollaborateswithagent_inv`, `concept_university_microsoft`
`concept_personus_steve_ballmer`, `concept:personleadsorganization`, `concept_university_microsoft`
`concept_personus_steve_ballmer`, `concept:worksfor`, `concept_university_microsoft`
`concept_personus_steve_ballmer`, `concept:proxyfor`, `concept_retailstore_microsoft`
`concept_personus_steve_ballmer`, `concept:subpartof`, `concept_retailstore_microsoft`
`concept_personus_steve_ballmer`, `concept:agentcontrols`, `concept_retailstore_microsoft`
`concept_person_bill`, `concept:personleadsorganization`, `concept_university_microsoft`
`concept_person_bill`, `concept:worksfor`, `concept_university_microsoft`
`concept_person_robbie_bach`, `concept:personleadsorganization`, `concept_university_microsoft`
`concept_person_robbie_bach`, `concept:worksfor`, `concept_university_microsoft`
`concept_retailstore_microsoft`, `concept:proxyfor_inv`, `concept_personus_steve_ballmer`
`concept_retailstore_microsoft`, `concept:subpartof_inv`, `concept_personus_steve_ballmer`
`concept_retailstore_microsoft`, `concept:agentcontrols_inv`, `concept_personus_steve_ballmer`

For the PersonBornInLocation task

Query: `(concept_person_mark001, concept:personborninlocation, concept_county_york_city)`

Selected key edges:

`concept_person_mark001`, `concept:persongraduatedfromuniversity`, `concept_university_college`
`concept_person_mark001`, `concept:persongraduatedschool`, `concept_university_college`
`concept_person_mark001`, `concept:persongraduatedfromuniversity`, `concept_university_state_university`
`concept_person_mark001`, `concept:persongraduatedschool`, `concept_university_state_university`
`concept_person_mark001`, `concept:personbornincity`, `concept_city_hampshire`
`concept_person_mark001`, `concept:hasspouse`, `concept_person_diane001`
`concept_person_mark001`, `concept:hasspouse_inv`, `concept_person_diane001`

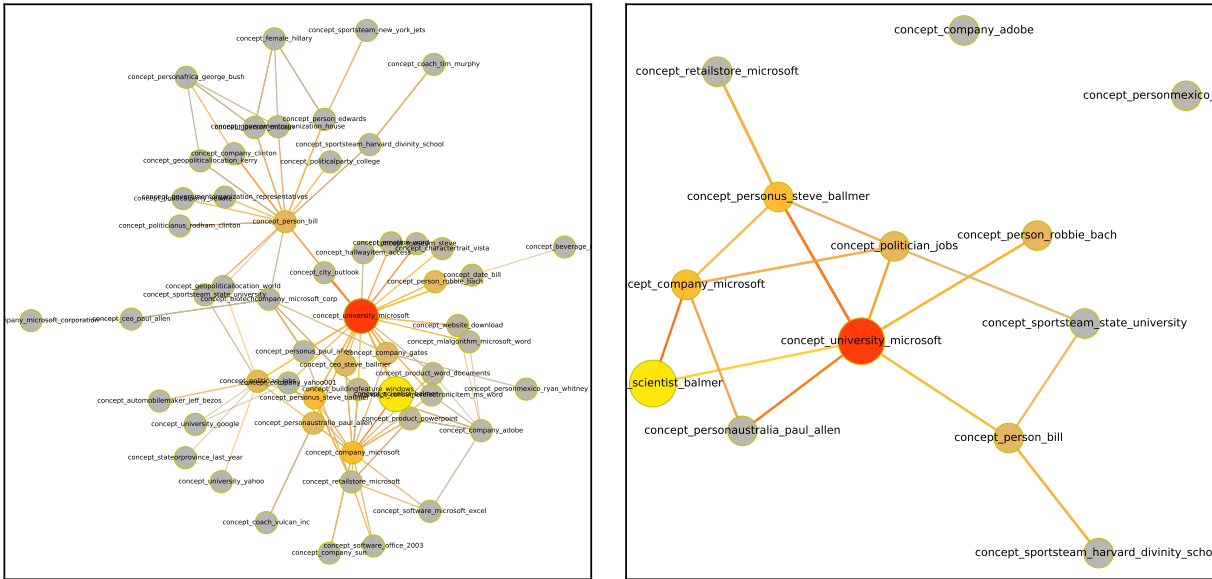


Figure 15: **WorksFor**. The head is *concept_scientist_balmer*, the query relation is *concept:worksfor*, and the desired tail is *concept_university_microsoft*. The left is a full subgraph derived with $max_attended_nodes_per_step = 20$, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

```

concept_university_college , concept:persongraduatedfromuniversity_inv , concept_person_mark001
concept_university_college , concept:persongraduatedschool_inv , concept_person_mark001
concept_university_college , concept:persongraduatedfromuniversity_inv , concept_person_bill
concept_university_college , concept:persongraduatedschool_inv , concept_person_bill
concept_university_state_university , concept:persongraduatedfromuniversity_inv , concept_person_mark001
concept_university_state_university , concept:persongraduatedschool_inv , concept_person_mark001
concept_university_state_university , concept:persongraduatedfromuniversity_inv , concept_person_bill
concept_university_state_university , concept:persongraduatedschool_inv , concept_person_bill
concept_city_hampshire , concept:personbornincity_inv , concept_person_mark001
concept_person_diane001 , concept:persongraduatedfromuniversity , concept_university_state_university
concept_person_diane001 , concept:persongraduatedschool , concept_university_state_university
concept_person_diane001 , concept:hasspouse , concept_person_mark001
concept_person_diane001 , concept:hasspouse_inv , concept_person_mark001
concept_person_diane001 , concept:personborninlocation , concept_county_york_city
concept_university_state_university , concept:persongraduatedfromuniversity_inv , concept_person_diane001
concept_university_state_university , concept:persongraduatedschool_inv , concept_person_diane001
concept_person_bill , concept:personbornincity , concept_city_york
concept_person_bill , concept:personborninlocation , concept_city_york
concept_person_bill , concept:persongraduatedfromuniversity , concept_university_college
concept_person_bill , concept:persongraduatedschool , concept_university_college
concept_person_bill , concept:persongraduatedfromuniversity , concept_university_state_university
concept_person_bill , concept:persongraduatedschool , concept_university_state_university
concept_city_york , concept:personbornincity_inv , concept_person_bill
concept_city_york , concept:personbornincity_inv , concept_person_diane001
concept_university_college , concept:persongraduatedfromuniversity_inv , concept_person_diane001
concept_person_diane001 , concept:personbornincity , concept_city_york

```

For the PersonLeadsOrganization task

Query: (concept_journalist_bill_plante , concept:personleadsorganization , concept_company_cnn_pbs)

Selected key edges:

```

concept_journalist_bill_plante , concept:worksfor , concept_televisionnetwork_cbs
concept_journalist_bill_plante , concept:agentcollaborateswithagent_inv , concept_televisionnetwork_cbs
concept_televisionnetwork_cbs , concept:worksfor_inv , concept_journalist_walter_cronkite
concept_televisionnetwork_cbs , concept:agentcollaborateswithagent , concept_journalist_walter_cronkite
concept_televisionnetwork_cbs , concept:worksfor_inv , concept_personus_scott_pelley
concept_televisionnetwork_cbs , concept:worksfor_inv , concept_actor_daniel_schorr

```

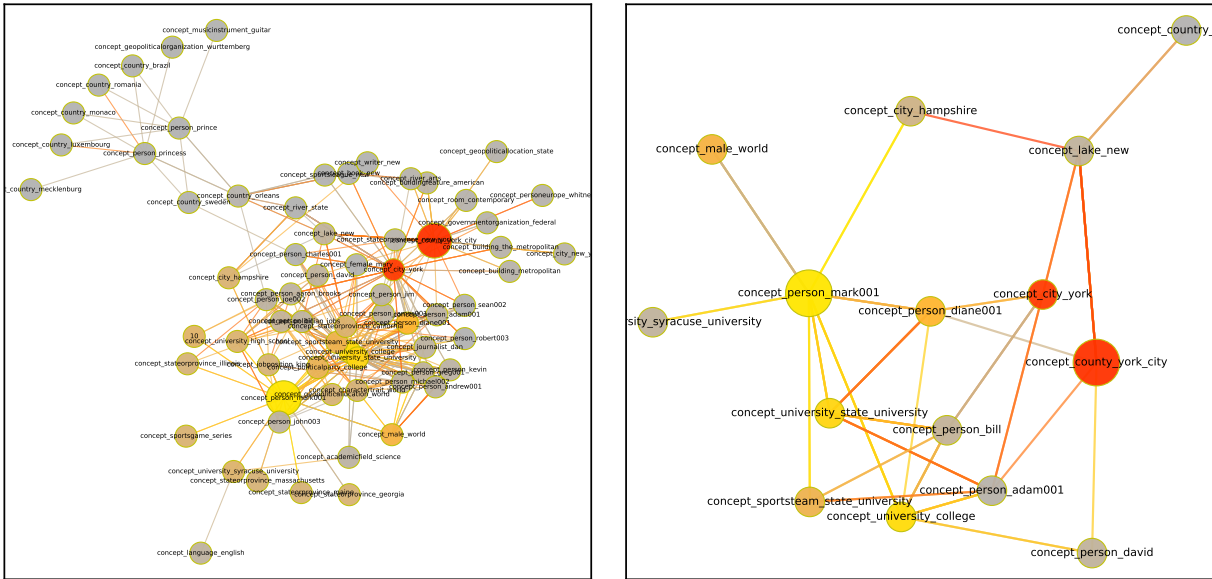


Figure 16: **PersonBornInLocation**. The head is `concept_person_mark001`, the query relation is `concept:personborninlocation`, and the desired tail is `concept_county_york_city`. The left is a full subgraph derived with $max_attended_nodes_per_step = 20$, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

```

concept_televisionnetwork_cbs, concept_worksfor_inv, concept_person_edward_r_murrow
concept_televisionnetwork_cbs, concept:agentcollaborateswithagent, concept_person_edward_r_murrow
concept_televisionnetwork_cbs, concept_worksfor_inv, concept_journalist_bill_plante
concept_televisionnetwork_cbs, concept:agentcollaborateswithagent, concept_journalist_bill_plante
concept_journalist_walter_cronkite, concept:worksfor, concept_televisionnetwork_cbs
concept_journalist_walter_cronkite, concept:agentcollaborateswithagent_inv, concept_televisionnetwork_cbs
concept_journalist_walter_cronkite, concept:worksfor, concept_nonprofitorganization_cbs_evening
concept_personus_scott_pelley, concept:worksfor, concept_televisionnetwork_cbs
concept_personus_scott_pelley, concept:personleadsorganization, concept_televisionnetwork_cbs
concept_personus_scott_pelley, concept:personleadsorganization, concept_company_cnn_pbs
concept_actor_daniel_schorr, concept:worksfor, concept_televisionnetwork_cbs
concept_actor_daniel_schorr, concept:personleadsorganization, concept_televisionnetwork_cbs
concept_actor_daniel_schorr, concept:personleadsorganization, concept_company_cnn_pbs
concept_person_edward_r_murrow, concept:worksfor, concept_televisionnetwork_cbs
concept_person_edward_r_murrow, concept:agentcollaborateswithagent_inv, concept_televisionnetwork_cbs
concept_person_edward_r_murrow, concept:personleadsorganization, concept_televisionnetwork_cbs
concept_person_edward_r_murrow, concept:personleadsorganization, concept_company_cnn_pbs
concept_televisionnetwork_cbs, concept:organizationheadquarteredin, concept_city_new_york
concept_televisionnetwork_cbs, concept:agentcollaborateswithagent, concept_personeuropewilliam_paley
concept_televisionnetwork_cbs, concept:topmemberoforganization_inv, concept_personeuropewilliam_paley
concept_company_cnn_pbs, concept:headquarteredin, concept_city_new_york
concept_company_cnn_pbs, concept:personbelongstoorganization_inv, concept_personeuropewilliam_paley
concept_nonprofitorganization_cbs_evening, concept:worksfor_inv, concept_journalist_walter_cronkite
concept_city_new_york, concept:organizationheadquarteredin, concept_televisionnetwork_cbs
concept_city_new_york, concept:headquarteredin_inv, concept_televisionnetwork_cbs
concept_city_new_york, concept:headquarteredin_inv, concept_company_cnn_pbs
concept_personeuropewilliam_paley, concept:agentcollaborateswithagent_inv, concept_televisionnetwork_cbs
concept_personeuropewilliam_paley, concept:topmemberoforganization, concept_televisionnetwork_cbs
concept_personeuropewilliam_paley, concept:personbelongstoorganization, concept_company_cnn_pbs
concept_personeuropewilliam_paley, concept:personleadsorganization, concept_company_cnn_pbs

```

For the OrganizationHiredPerson task

Query: `(concept_stateorprovince_afternoon, concept:organizationhiredperson, concept_personmexico_ryan_whitney)`

Selected key edges:

```
concept_stateorprovince_afternoon, concept:atdate, concept_dateliteral_n2007
```

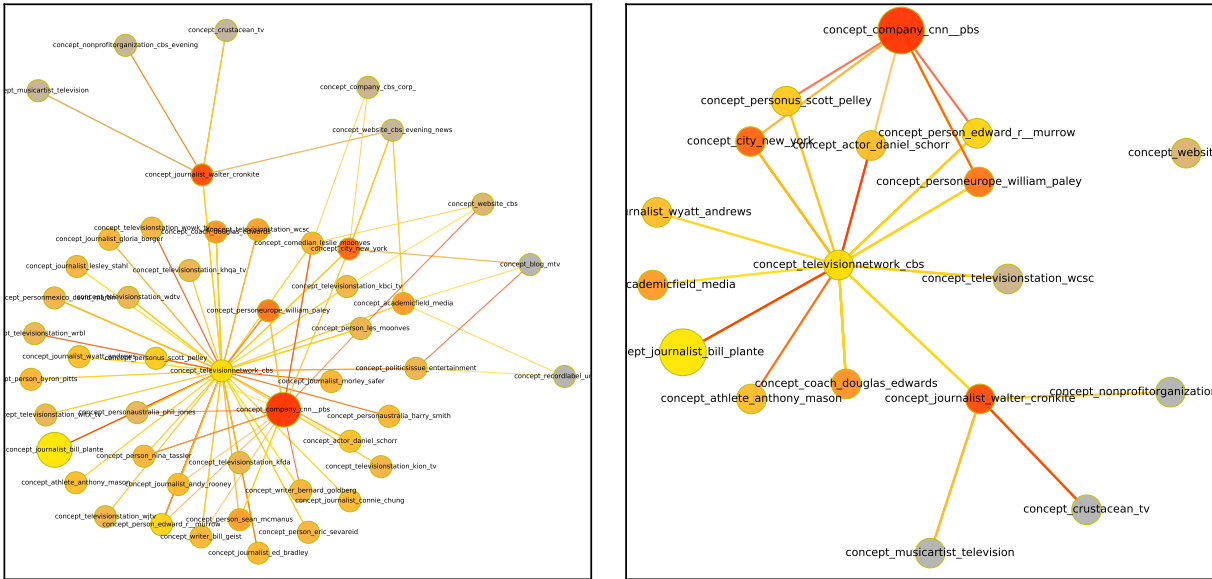


Figure 17: **PersonLeadsOrganization**. The head is *concept_journalist_bill_plante*, the query relation is *concept:organizationheadquarteredincity*, and the desired tail is *concept_company_cnn_pbs*. The left is a full subgraph derived with *max_attended_nodes_per_step* = 20, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

```

concept_stateorprovince_afternoon, concept:atdate, concept_date_n2003
concept_stateorprovince_afternoon, concept:atdate, concept_dateliteral_n2006
concept_dateliteral_n2007, concept:atdate_inv, concept_country_united_states
concept_dateliteral_n2007, concept:atdate_inv, concept_city_home
concept_dateliteral_n2007, concept:atdate_inv, concept_city_service
concept_dateliteral_n2007, concept:atdate_inv, concept_country_left_parties
concept_date_n2003, concept:atdate_inv, concept_country_united_states
concept_date_n2003, concept:atdate_inv, concept_city_home
concept_date_n2003, concept:atdate_inv, concept_city_service
concept_date_n2003, concept:atdate_inv, concept_country_left_parties
concept_dateliteral_n2006, concept:atdate_inv, concept_country_united_states
concept_dateliteral_n2006, concept:atdate_inv, concept_city_home
concept_dateliteral_n2006, concept:atdate_inv, concept_city_service
concept_dateliteral_n2006, concept:atdate_inv, concept_country_left_parties
concept_country_united_states, concept:atdate, concept_year_n1992
concept_country_united_states, concept:atdate, concept_year_n1997
concept_country_united_states, concept:organizationhireperson, concept_personmexico_ryan_whitney
concept_city_home, concept:atdate, concept_year_n1992
concept_city_home, concept:atdate, concept_year_n1997
concept_city_home, concept:organizationhireperson, concept_personmexico_ryan_whitney
concept_city_service, concept:atdate, concept_year_n1992
concept_city_service, concept:atdate, concept_year_n1997
concept_city_service, concept:organizationhireperson, concept_personmexico_ryan_whitney
concept_country_left_parties, concept:worksfor_inv, concept_personmexico_ryan_whitney
concept_country_left_parties, concept:organizationhireperson, concept_personmexico_ryan_whitney
concept_year_n1992, concept:atdate_inv, concept_governmentorganization_house
concept_year_n1992, concept:atdate_inv, concept_country_united_states
concept_year_n1992, concept:atdate_inv, concept_city_home
concept_year_n1992, concept:atdate_inv, concept_tradeunion_congress
concept_year_n1997, concept:atdate_inv, concept_governmentorganization_house
concept_year_n1997, concept:atdate_inv, concept_country_united_states
concept_year_n1997, concept:atdate_inv, concept_city_home
concept_personmexico_ryan_whitney, concept:worksfor, concept_governmentorganization_house
concept_personmexico_ryan_whitney, concept:worksfor, concept_tradeunion_congress
concept_personmexico_ryan_whitney, concept:worksfor, concept_country_left_parties
concept_governmentorganization_house, concept:personbelongstoorganization_inv, concept_personus_party
concept_governmentorganization_house, concept:worksfor_inv, concept_personmexico_ryan_whitney
concept_governmentorganization_house, concept:organizationhireperson, concept_personmexico_ryan_whitney

```

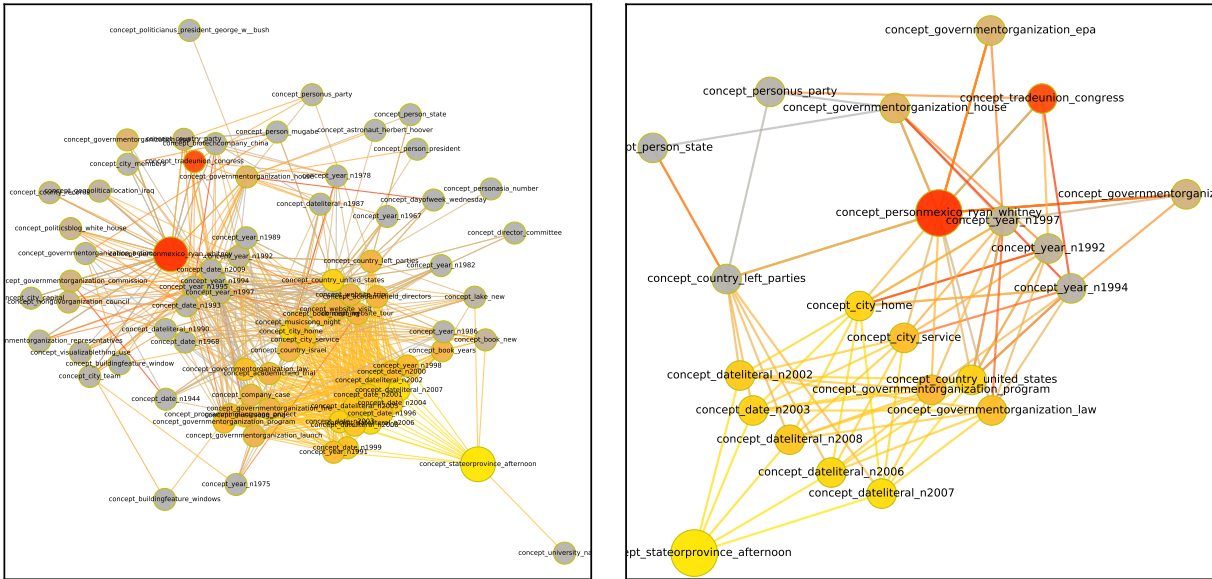


Figure 18: **OrganizationHiredPerson**. The head is *concept_stateorprovince_afternoon*, the query relation is *concept:organizationhiredperson*, and the desired tail is *concept_personmexico_ryan_whitney*. The left is a full subgraph derived with *max_attended_nodes_per_step* = 20, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

concept_tradeunion_congress, concept:organizationhiredperson, concept_personus_party
 concept_tradeunion_congress, concept:worksfor_inv, concept_personmexico_ryan_whitney
 concept_tradeunion_congress, concept:organizationhiredperson, concept_personmexico_ryan_whitney
 concept_country_left_parties, concept:organizationhiredperson, concept_personus_party

For the AgentBelongsToOrganization task

Query: (concept_person_mark001, concept:agentbelongstoorganization, concept_geopoliticallocation_world)

Selected key edges:

concept_person_mark001, concept:personbelongstoorganization, concept_sportsteam_state_university
 concept_person_mark001, concept:agentcollaborateswithagent, concept_male_world
 concept_person_mark001, concept:agentcollaborateswithagent_inv, concept_male_world
 concept_person_mark001, concept:personbelongstoorganization, concept_politicalparty_college
 concept_sportsteam_state_university, concept:personbelongstoorganization_inv, concept_politician_jobs
 concept_sportsteam_state_university, concept:personbelongstoorganization_inv, concept_person_mark001
 concept_sportsteam_state_university, concept:personbelongstoorganization_inv, concept_person_greg001
 concept_sportsteam_state_university, concept:personbelongstoorganization_inv, concept_person_michael002
 concept_male_world, concept:agentcollaborateswithagent, concept_politician_jobs
 concept_male_world, concept:agentcollaborateswithagent_inv, concept_politician_jobs
 concept_male_world, concept:agentcollaborateswithagent, concept_person_mark001
 concept_male_world, concept:agentcollaborateswithagent_inv, concept_person_mark001
 concept_male_world, concept:agentcollaborateswithagent, concept_person_greg001
 concept_male_world, concept:agentcollaborateswithagent_inv, concept_person_greg001
 concept_male_world, concept:agentcontrols, concept_person_greg001
 concept_male_world, concept:agentcollaborateswithagent, concept_person_michael002
 concept_male_world, concept:agentcollaborateswithagent_inv, concept_person_michael002
 concept_politicalparty_college, concept:personbelongstoorganization_inv, concept_person_mark001
 concept_politicalparty_college, concept:personbelongstoorganization_inv, concept_person_greg001
 concept_politicalparty_college, concept:personbelongstoorganization_inv, concept_person_michael002
 concept_politician_jobs, concept:personbelongstoorganization, concept_sportsteam_state_university
 concept_politician_jobs, concept:agentcollaborateswithagent, concept_male_world
 concept_politician_jobs, concept:agentcollaborateswithagent_inv, concept_male_world
 concept_politician_jobs, concept:worksfor, concept_geopoliticallocation_world
 concept_person_greg001, concept:personbelongstoorganization, concept_sportsteam_state_university
 concept_person_greg001, concept:agentcollaborateswithagent, concept_male_world

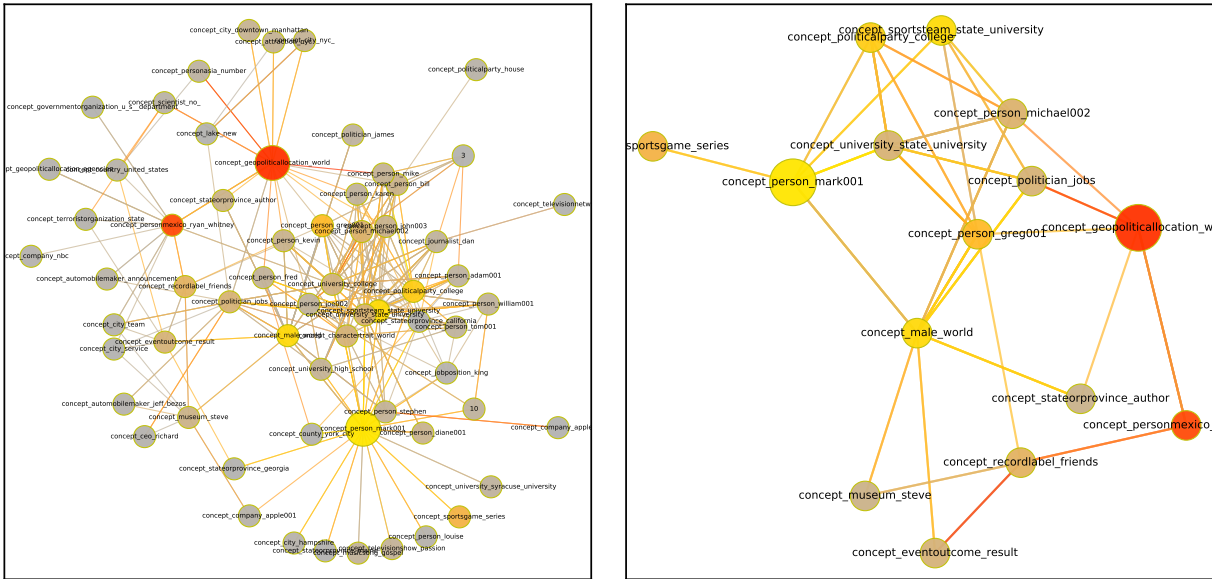


Figure 19: **AgentBelongsToOrganization**. The head is `concept_person_mark001`, the query relation is `concept:agentbelongstoorganization`, and the desired tail is `concept_geopoliticalallocation_world`. The left is a full subgraph derived with `max_attended_nodes_per_step = 20`, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.

```

concept_person_greg001, concept:agentcollaborateswithagent_inv, concept_male_world
concept_person_greg001, concept:agentcontrols_inv, concept_male_world
concept_person_greg001, concept:agentbelongstoorganization, concept_geopoliticalallocation_world
concept_person_greg001, concept:personbelongstoorganization, concept_politicalparty_college
concept_person_greg001, concept:agentbelongstoorganization, concept_recordlabel_friends
concept_person_michael002, concept:personbelongstoorganization, concept_sportsteam_state_university
concept_person_michael002, concept:agentcollaborateswithagent, concept_male_world
concept_person_michael002, concept:agentcollaborateswithagent_inv, concept_male_world
concept_person_michael002, concept:agentbelongstoorganization, concept_geopoliticalallocation_world
concept_person_michael002, concept:personbelongstoorganization, concept_politicalparty_college
concept_geopoliticalallocation_world, concept:worksfor_inv, concept_personmexico_ryan_whitney
concept_geopoliticalallocation_world, concept:organizationhiredperson, concept_personmexico_ryan_whitney
concept_recordlabel_friends, concept:organizationhiredperson, concept_personmexico_ryan_whitney
concept_personmexico_ryan_whitney, concept:worksfor, concept_geopoliticalallocation_world
concept_personmexico_ryan_whitney, concept:organizationhiredperson_inv, concept_geopoliticalallocation_world
concept_personmexico_ryan_whitney, concept:organizationhiredperson_inv, concept_recordlabel_friends

```

For the TeamPlaysInLeague task

Query: `(concept_sportsteam_mavericks, concept:teamplaysinleague, concept_sportsleague_nba)`

Selected key edges:

```

concept_sportsteam_mavericks, concept:teamplayssport, concept_sport_basketball
concept_sportsteam_mavericks, concept:teamplaysagainstteam, concept_sportsteam_boston_celtics
concept_sportsteam_mavericks, concept:teamplaysagainstteam_inv, concept_sportsteam_boston_celtics
concept_sportsteam_mavericks, concept:teamplaysagainstteam, concept_sportsteam_spurs
concept_sportsteam_mavericks, concept:teamplaysagainstteam_inv, concept_sportsteam_spurs
concept_sport_basketball, concept:teamplayssport_inv, concept_sportsteam_college
concept_sport_basketball, concept:teamplayssport_inv, concept_sportsteam_marshall_university
concept_sportsteam_boston_celtics, concept:teamplaysinleague, concept_sportsleague_nba
concept_sportsteam_spurs, concept:teamplaysinleague, concept_sportsleague_nba
concept_sportsleague_nba, concept:agentcompeteswithagent, concept_sportsleague_nba
concept_sportsleague_nba, concept:agentcompeteswithagent_inv, concept_sportsleague_nba
concept_sportsteam_college, concept:teamplaysinleague, concept_sportsleague_international

```

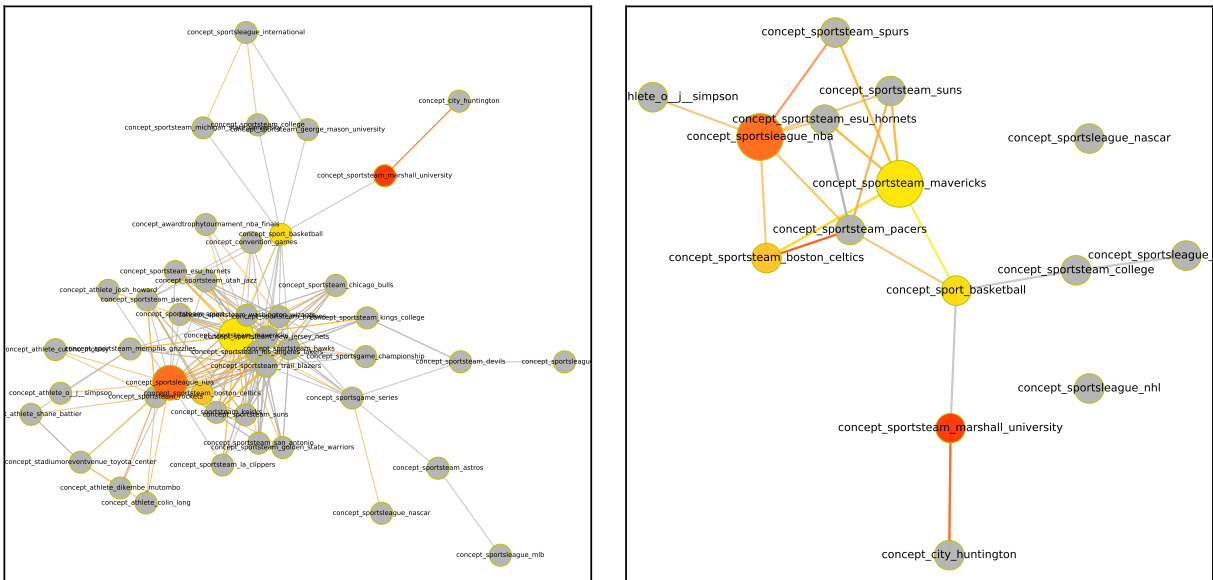


Figure 20: **TeamPlaysInLeague**. The head is *concept_sportsteam_mavericks*, the query relation is *concept:teamplaysinleague*, and the desired tail is *concept_sportsleague_nba*. The left is a full subgraph derived with $max_attended_nodes_per_step = 20$, and the right is a further extracted subgraph from the left based on attention. The big yellow node represents the head, and the big red node represents the tail. Colors indicate how important a node is attended to in a local subgraph. Grey means less important, yellow means it is more attended during the early steps, and red means it is more attended when getting close to the final step.