CoRGi: Content-Rich Graph Neural Networks with Attention

JOOYEON KIM*, RIKEN, Japan
ANGUS LAMB, Microsoft Research Cambridge, UK, UK
SIMON WOODHEAD, Eedi, UK
SIMON PEYTON JONES, CHENG ZHANG, and MILTIADIS ALLAMANIS, Microsoft Research, UK

Graph representations of a target domain often project it to a set of entities (nodes) and their relations (edges). However, such projections often miss important and rich information. For example, in graph representations used in missing value imputation, items — represented as nodes — may contain rich textual information. However, when processing graphs with graph neural networks (GNN), such information is either ignored or summarized into a single vector representation used to initialize the GNN. Towards addressing this, we present CoRGi, a GNN that considers the rich data within nodes in the context of their neighbors. This is achieved by endowing CoRGi's message passing with a personalized attention mechanism over the content of each node. This way, CoRGi assigns *user-item-specific* attention scores with respect to the words that appear in items. We evaluate CoRGi on two edge-value prediction tasks and show that CoRGi is better at making edge-value predictions, especially on sparse regions of the graph.

Additional Key Words and Phrases: graph neural networks, recommendations

ACM Reference Format:
Jooyeon Kim, Angus Lamb, Simon Woodhead, Simon Peyton Jones, Cheng Zhang, and Miltiadis Allamanis. 2021. CoRGi: Content-Rich Graph Neural Networks with Attention. 1, 1 (September 2021), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Graph neural networks (GNN) [20, 22, 43] have enjoyed great success in deep learning research. GNNs allow us to model complex graph-structured data. While graph representations of data may be reasonable, the construction of the input graphs is often a lossy projection of the data of the modeled domain. For example, a graph representation of a book recommendation problem may represent books and users as nodes with recommendations acting as valued edges. However, each book-node contains rich semi-structured content, such as text structured into sections, figures, tables, etc., which can be used to improve the performance of recommendations.

A common approach to incorporate the content of nodes in GNNs is to "summarize" it into a single vector representation (embedding) that compresses all relevant information. This often includes computing a single vector representation from a whole sentence or document using an encoder model, such as a transformer or a simpler bag-of-words model. However, such representations are suboptimal, given the relatively small size of these vectors and that they are pre-computed outside of the application context. This is recognized in the literature in natural language processing

*Work done while at Microsoft Research.

Authors' addresses: Jooyeon Kim, trovato@corporation.com, RIKEN, Tokyo, Japan; Angus Lamb, Microsoft Research Cambridge, UK, 21 Station Road, Cambridge, UK, larst@microsoft.com; Simon Woodhead, simon.woodhead@eedi.co.uk, Eedi, UK; Simon Peyton Jones, simonpj@microsoft.com; Cheng Zhang, chezha@microsoft.com; Miltiadis Allamanis, miallama@microsoft.com, Microsoft Research, 21 Station Road, Cambridge, UK.

(NLP). Instead of representing inputs as a single vector/embedding, the full input is used: while generating the output, encoder-decoder models employ some form of attention mechanism over the whole input given the model context instead of representing it as a single vector. For example, in NLP text summarization [61] a decoder attends to the encoded representations of all the words in the input text.

In the same fashion, we need a better way for a GNN to capture the content within nodes of a graph. Towards this goal, we present CoRGi (Content-Rich Graph neural network with attention), a message-passing GNN [11, 20, 22] that incorporates an attention mechanism over the rich content of each node when computing edge representations. This allows CoRGi to effectively learn *both* about the structure of the data and the content within each node.

One application of CoRGi is edge-value imputation, e.g., missing value imputation with GNNs in collaborative filtering [60] (Figure 1-left). For example, in a dataset of student-question answers, each question is associated with a rich textual description. A graph-based representation allows capturing the rich interactions among students and questions (user responses) but would ignore important content within items (textual descriptions of questions; text in Figure 1). CoRGi combines both sources of information through a personalized attention mechanism that yields a user-item pair-specific (student-question-specific) representation of the item's content. In section 4 we show that CoRGi achieves better performance than baselines, and especially in sparse regions of the user-item graph, such as rarely matched items. We believe that CoRGi's hybrid approach of processing both user response data and content is beneficial across a wide range of collaborative filtering and recommender systems and other uses of GNNs.

Contributions. In summary, our contributions are

- CoRGi: a message-passing GNN which incorporates an attention mechanism over node content when computing a message over each edge (section 2).
- We specialize CoRGi for the user-item recommendation tasks (subsection 2.1).
- In an extensive evaluation (section 4) over two real-world datasets we show that — compared to baselines — CoRGi can improve user-response prediction performance, especially for items with few user ratings, where content plays a critical role.

2 CORGI: GNNS WITH ATTENTION OVER NODE CONTENT

In this section, we first describe the problem setting for CoRGi and then discuss its implementation. We then focus on the usecase of recommender systems with textual content in item nodes. Finally, we discuss the computational complexity of CoRGi and discuss strategies to reduce the time complexity and memory requirements.

In Table 1, we summarize the key notations used throughout the paper. We group notations in three groups: (a) notations on graph sets and the corresponding elements. (b) variables and parameters used to describe the forward pass of CoRGi. (c) notations used to describe the content information associated to content or item nodes. In addition to this, we have the trainable weights explained during the message passing of CoRGi: $\mathbf{P}^{(l)}$ and $\mathbf{Q}^{(l)}$ for updating node embeddings, $\mathbf{W}_U^{(l)}$, $\mathbf{W}_M^{(l)}$, and $\mathbf{p}^{(l)}$ for computing attention coefficients, and $\mathbf{w}_{\text{out}}$ and b for the prediction MLP.

Problem setting. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v \in \mathcal{V}$ is associated with node features $\mathbf{h}_v^{(0)}$ and each edge with the features $\mathbf{e}_{ij}^{(0)}, \forall (i,j) \in \mathcal{E}$. If a node or edge is *not* related with any features a constant value may be assigned. A subset of nodes $\mathcal{V}_C \subset \mathcal{V}$ is associated with a set of $n(i)$ *content* vector representations $\mathbf{Z}_i = \{\mathbf{z}_1^{(i)}, \ldots, \mathbf{z}_{n(i)}^{(i)}\}, \forall v_i \in \mathcal{V}_C$, and $\mathbf{z}_k^{(i)} \in \mathbb{R}^D$. Notice that $n(i)$, the number of content vectors associated per each node v_i, may differ by nodes. The $\mathbf{z}_k^{(i)}$ may be direct inputs to the model or the outputs of another deep learning component that encodes the content

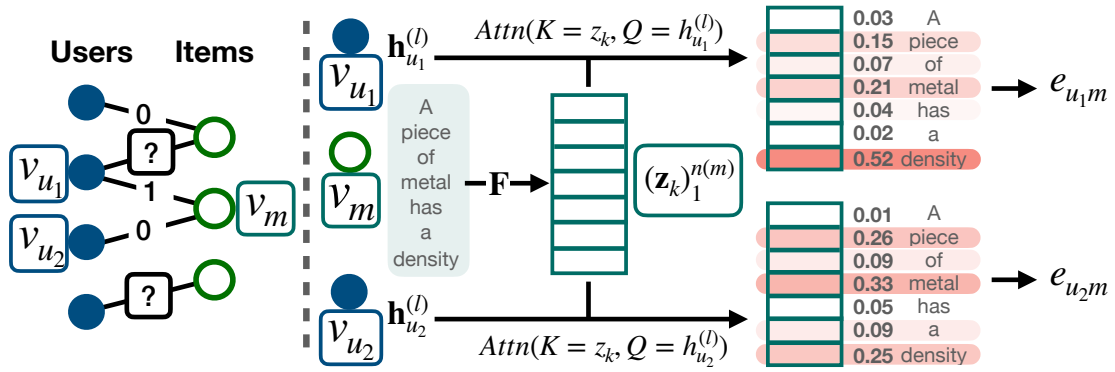


Fig. 1. In the educational setting, students (users) and questions (items) form a bipartite user-item graph (left) and predicting student responses can be posed as a missing edge imputation problem. The value of an edge is known for some user-item pairs but not for all. The edge representation computation in a GNN message-passing layer l (right) for CoRGi. Item nodes are associated with content ((e.g., a question v_m) contains text as in this figure). By encoding the content through a model $\mathbf{F}$ (e.g., through a transformer), CoRGi obtains a set of content vectors $\{\mathbf{z}_k\}_1^{n(m)}$. The embedding of an edge (e.g., $\mathbf{e}_{u_1 m}$) is computed by using the node embedding of a user (student) node (e.g., $\mathbf{h}_{u_1}$, $\mathbf{h}_{u_2}$) and an attention mechanism over $\{\mathbf{z}_k\}_1^{n(m)}$. This computes an edge representation that takes into consideration the content within the item (question) nodes in a personalized user-dependent way.

Table 1. Key notations used in the paper.

	Symbols	Description
Graph sets & elements	$\mathcal{V}$	The set of all nodes in the graph
	$\mathcal{V}_C, \mathcal{V}_M, \mathcal{V}_U \subset \mathcal{V}$	The sets of content, item, and user nodes
	$\mathcal{E}$	The set of all edges in the graph
	$\mathcal{N}(i)$	Neighborhood function for node v_i
CoRGi variables	$\mathbf{h}_i^{(0)}$	Input feature of node v_i of size $C^{(l,h)}$
	$\mathbf{e}_{ij}^{(0)}$	Input feature of edge e_{ij} of size $C^{(l,e)}$
	$\mathbf{h}_i^{(l)}$	Node embedding of v_i at l^{th} layer
	$\mathbf{e}_{ij}^{(l)}$	Edge embedding between v_i and v_j at l^{th} layer
	$\mathbf{e}_{ij}^{(l)\prime}$	Edge embedding before content update
	$\mathbf{e}_{ij,\text{CA}}^{(l)}$	Edge embedding from content-attention
	$c_{ik}^{(l)}$	Attention coefficient between v_i and content k
	$\alpha_{ik}^{(l)}$	Attention probability from $c_{ik}^{(l)}$ after Softmax
Content-related notations	$n(i)$	The number of content vectors associated to v_i
	$\mathbf{Z}_i = \{\mathbf{z}_k^{(i)}\}_1^{n(i)} \subset \mathbb{R}^D$	A set of content vectors associated to v_i
	$\mathbf{D}_i = [w_1^{(i)}, \cdots, w_{n(i)}^{(i)}]$	A sequence of words associated to v_i
	$\mathbf{F}$	A sequence encoder that projects $\mathbf{D}_i$ to $\mathbf{Z}_i$

(e.g., a transformer [50]). The goal of CoRGi is to learn representations over the nodes and edges while considering the set of content vectors within each node.

Algorithm 1: CoRGi forward computation for user-response prediction

Input: Bi-partite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \mathcal{V}_U \cup \mathcal{V}_M$; node features $\mathbf{h}_v^{(0)}, \forall v \in \mathcal{V}$ and edge attributes $\mathbf{e}_{ij}^{(0)}, \forall (i,j) \in \mathcal{E}$; number of layers L; content sequence encoder $\mathbf{F}$; Weight matrices and vectors $\mathbf{P}^{(l)}$ for message passing, $\mathbf{Q}^{(l)}$ for node updating, $\mathbf{W}_U^{(l)}$, $\mathbf{W}_M^{(l)}$ and $\mathbf{p}^{(l)}$ for computing attention coefficients between users U and items M; non-linearity σ; aggregation functions AGG_l; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$; sampled item nodes $\mathcal{V}'_m$ from the neighbor sampler

for $l \in \{1, \ldots, L\}$ **do**
  for $i \in \mathcal{V}$ **do**
    $\mathbf{m}_{ij}^{(l)} \leftarrow$ Compute messages from Equation 1 $\quad \forall j \in \mathcal{N}(i)$;
    $\mathbf{h}_i^{(l)} \leftarrow$ Update node states with Equation 2 ;
    for $v_j \in \mathcal{N}(i)$ **do**
      $c_{ik}^{(l)} \leftarrow$ Compute content attention scores from Equation 6 ;
      $\alpha_{ik}^{(l)} \leftarrow$ Compute content attention probability from Equation 5 ;
      $\mathbf{e}_{ij}^{(l)\prime} \leftarrow$ Update content-independent edge embedding with Equation 3 ;
      $\mathbf{e}_{ij,\text{CA}}^{(l)} \leftarrow$ Update content-attention edge embedding with Equation 4 or use cache;
      $\mathbf{e}_{ij}^{(l)} \leftarrow \mathbf{e}_{ij}^{(l)\prime} + \mathbf{e}_{ij,\text{CA}}^{(l)}$

Output: Node embeddings $\mathbf{h}_v, \forall v \in \mathcal{V}$

CoRGi. learns node and edge embeddings using the graph and the content information within nodes. (Figure 1) It generally follows the message-passing GNN paradigm [11] and is closely related to GRAPE [60]. However, in contrast to existing models, CoRGi uses the content vector representations associated with each node during message-passing with personalized attention. Specifically, CoRGi computes message by learning to focus on potentially different parts of the content in the context of the neighboring nodes using an attention mechanism (Figure 1). Algorithm 1 presents a high-level overview of CoRGi which we discuss next.

CoRGi's architecture assume L message passing layers, similar to most GNNs [20, 22]. Following the construction of You et al. [60], at each layer l, CoRGi computes a *message* $\mathbf{m}_{ij}^{(l)}$ from node v_j to v_i using the previous-level node embedding $\mathbf{h}_i^{(l-1)}$ and edge embedding $\mathbf{e}_{ij}^{(l-1)}$

$$\mathbf{m}_{ij}^{(l)} = \sigma\left(\mathbf{P}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_j^{(l-1)}, \mathbf{e}_{ij}^{(l-1)}\right)\right), \tag{1}$$

where σ is a non-linearity and $\mathbf{P}^{(l)}$ is a trainable weight. We set $\mathbf{h}_i^{(0)}$ and $\mathbf{e}_{ij}^{(0)}$ to the input node features and input edge attributes (if any), and $\mathbf{h}^{(l)} \in \mathbb{R}^{C^{(l,h)}}$, $\mathbf{e}^{(l)} \in \mathbb{R}^{C^{(l,e)}}$. We then *aggregate* messages from all neighbors of v_i and *update* the node embedding with a learnable weight $\mathbf{Q}^{(l)}$:

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{Q}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_i^{(l-1)}, \text{AGG}^{(l)}\left(\mathbf{m}_{ij}^{(l)} \mid \forall j \in \mathcal{N}(i)\right)\right)\right), \tag{2}$$

where $\text{AGG}^{(l)}$ is a permutation-invariant aggregation function. $\mathcal{N}(i)$ is v_i's set of neighborhoods.

So far, the message-passing in CoRGi is identical to that of You et al. [60]. However, we are also interested in incorporating information from the content of each node $v_j \in \mathcal{V}_C$. We achieve this through an attention mechanism within the GNN-message passing. This allows a message between a v_i and v_j to *focus* on a specific part of the content.

Such an ability can be helpful in many scenarios. For example, in an educational recommender system, the attended (textual) content of a question is an essential factor in predicting the student's ability to answer it correctly given, e.g., a diagnostic question [54]. Intuitively, a student — given her skills — may focus on different aspects of the question when answering it. The location of focus will vary across students with different skills, where the student learned the topics correctly may focus on the key information while the student with misconceptions may focus on the distractor parts of such diagnostic questions. CoRGi's attention mechanism aims to emulate this effect.

In CoRGi we model this by combining any edge features $\mathbf{e}_{ij}^{(0)}$ with a content-attention vector $\mathbf{e}_{ij,\text{CA}}^{(l)}$ computed from an attention mechanism over the content $\mathbf{Z}_{v_j}$. While there are many possible design options, we consider two options. (a) an element-wise addition $\mathbf{e}_{ij}^{(l)} = \mathbf{e}_{ij}^{(l)\prime} + \mathbf{e}_{ij,\text{CA}}^{(l)}$, or (b) a concatenation operation $\mathbf{e}_{ij}^{(l)} = \textsc{Concat}(\mathbf{e}_{ij}^{(0)}, \mathbf{e}_{ij,\text{CA}}^{(l)})$, where

$$\mathbf{e}_{ij}^{(l)\prime} = \sigma\big(\mathbf{W}^{(l)} \cdot \textsc{Concat}(\mathbf{h}_j^{(l)}, \mathbf{e}_{ij}^{(0)})\big), \tag{3}$$

with a trainable weight $\mathbf{W}^{(l)}$, and $\mathbf{e}_{ij,\text{CA}}^{(l)}$ is information computed by the attention mechanism. Equation 3 is similar to the one used by You et al. [60]. Note that for the elementwise addition $\mathbf{e}_{ij}^{(l)\prime}$ has to have the same cardinality as $\mathbf{e}_{ij,\text{CA}}^{(l)}$.

We compare the predictive performance with examples of computed attention distributions using the element-wise operation and concatenation in section 4.2

Finally, we describe the attention mechanism yielding $\mathbf{e}_{ij,\text{CA}}^{(l)}$. For an edge between v_i and v_j at l^{th} layer, the content-attention (CA) is computed using the set of content vector representations of v_j, $\mathbf{Z}_j$, and the previous-level node embedding $\mathbf{h}_i^{(l-1)}$, i.e.,

$$\mathbf{e}_{ij,\text{CA}}^{(l)} = \textsc{Attention}\left(\textsc{keys} = \mathbf{Z}_j,\ \textsc{query} = \mathbf{h}_i^{(l-1)}\right) = \sum_k \alpha_{ik} \mathbf{W}_M^{(l)} \mathbf{z}_k^{(v_j)} \tag{4}$$

where $\mathbf{W}_M^{(l)}$ is a trainable weight, and α_{ik} is the attention probability computed as

$$\alpha_{ik}^{(l)} = \textsc{SoftMax}_k\left(c_{ik}^{(l)} \ \mid\ \forall k \in \{1, \ldots, n(i)\}\right), \tag{5}$$

i.e., the softmax over the attention scores $c_{ik}^{(l)}$. We test two commonly used attention mechanisms for computing $c_{ik}^{(l)}$: concatenation (CO) and dot-product (DP), computed as

$$c_{ik,CO}^{(l)} = \mathbf{p}^{(l)\top} \textsc{Concat}(\mathbf{W}_U^{(l)} \mathbf{h}_i^{(l-1)}, \mathbf{W}_M^{(l)} \mathbf{z}_k) \quad \text{and} \quad c_{ik,DP}^{(l)} = \left[\mathbf{W}_U^{(l)} \mathbf{h}_i^{(l-1)}\right]^\top \mathbf{W}_M^{(l)} \mathbf{z}_k, \tag{6}$$

where $\mathbf{W}_U^{(l)}$, and $\mathbf{p}^{(l)}$ are learnable weights.

Content representations. So far, we assumed that the content representation vectors $\mathbf{z}_j^{(v_i)} \in \mathbb{R}^D$ are given. In practice these representations can be computed from some deep learning component $\mathbf{F}$. CoRGi does *not* impose a structure on $\mathbf{F}$. For example, if the node content are images, then CNN-based architectures for $\mathbf{F}$ would be reasonable. Similarly, if the content is text, i.e., a sequence of words (or any other sequence), then any NLP model can be used. This includes text representation models [4, 28, 35], and sequence encoders [6, 37, 46] including transformers [8, 50]. Such models "contextualize" each individual word in the sequence and convert it to a *set* of vector representations.

2.1 CoRGi for user-response prediction

User response prediction is a common use case of GNNs and can be thought as an edge-value prediction task [2, 53, 60, 63]. We describe how we operationalize CoRGi for recommender systems. We consider a *bipartite* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with two disjoint node sets $\mathcal{V} = \mathcal{V}_U \cup \mathcal{V}_M$ of users and items. Each item node $v_m \in \mathcal{V}_M$ contains text $\mathbf{D}_m = [w_1^{(m)}, \cdots, w_{n(m)}^{(m)}]$,

i.e., a sequence of words. The sequence is then converted to a set of content vectors $\mathbf{Z}_m$ using a sequence encoder (e.g. a transformer) and is input to CoRGi.

An edge-level prediction (i.e., a recommendation $r(v_m, v_u)$ between a user v_u and an item v_m) between a user node v_u and an item node v_m can be made by concatenating the output user and item node embeddings passed through a final layer, i.e.,

$$r(v_m, v_u) = \sigma\left(\mathbf{w}_{\text{out}}^T \textsc{Concat}(\mathbf{h}_u^{(L)}, \mathbf{h}_m^{(L)}) + b\right), \tag{7}$$

where $\mathbf{w}_{\text{out}}$ and b are learnable weights.

2.2 Complexity analysis

CoRGi's computational and memory complexity is similar to other message-passing GNNs, with the additional cost from the attention mechanism. Compared to the node-to-node attention computation in GAT, CoRGi's attention mechanism involves maximum of $T = \max_{v \in \mathcal{V}_M}(|\mathbf{Z}_v|)$, the maximum content size with respect to $v \in \mathcal{V}_C$, for each content node. For the l^{th} message-passing CoRGi layer the computational complexity for computing the content attention is expressed as:

$$O\left(|\mathcal{V}_U| \cdot C^{(l-1,h)} \cdot C^{(l,e)} + |\mathcal{V}_M| \cdot T \cdot D \cdot C^{(l,e)} + |\mathcal{E}| \cdot T \cdot C^{(l,e)}\right).$$

The first and the second terms arise from the multiplication between the trainable weights and the node embeddings or content vector representations, in Equation 6. The last term arises during the pairwise linear operation in the attention coefficient calculation between the query and the key, in Equation 4.

We can drastically reduce the complexity by using the neighbour sampling method proposed by Hamilton et al. [13] and applying a caching trick for all $\mathbf{e}_{ij,\text{CA}}^{(l)}$. For network sampling, we sample a subset of nodes $\mathcal{V}' = \{\mathcal{V}'_U \cup \mathcal{V}'_M\}$ for the neighbor sampling and only update $\mathbf{e}_{ij,\text{CA}}$ whose target node v_j is in $\mathcal{V}'_M$ and source node v_i is in $\mathcal{N}(\mathcal{V}'_M)$. The sampled subgraph is $\mathcal{G}' = \{\mathcal{V}'', \mathcal{E}'\}$, with $\mathcal{V}'' = \{\mathcal{V}'_M \cup \mathcal{N}(\mathcal{V}'_M)\}$. This way, the computational cost is reduced.

Computing the attention for each layer is costly both in terms of memory and computation. To drastically reduce the memory and compute requirements, we use a caching trick for all $\mathbf{e}_{ij,\text{CA}}^{(l)}$. Since these representations can be thought as edge features, we want to compute them infrequently and re-use them. To do this, we create a cache for all $\mathbf{e}_{ij,\text{CA}}^{(l)}$ and initialize them with zeros. Then, at the final layer L, we compute $\mathbf{e}_{ij,\text{CA}}^{(L)}$ using Equation 4 and update the cache for all $\mathbf{e}_{ij,\text{CA}}^{(l)}$ to the computed $\mathbf{e}_{ij,\text{CA}}^{(L)}$. The newly cached values will be used in subsequent message-passing iterations. In this way, we avoid $L-1$ computations of Equation 4.

3 RELATED WORK

CoRGi is at the intersection of GNNs and machine learning models for missing value imputation. In this section, we discuss related missing value imputation models and message-passing GNNs for recommender systems. Models that used as baselines in section 4 are *italicized*.

Missing value imputation is the task of filling in previously unknown entries with predicted values. For two heterogeneous groups, namely, users and items, the task is commonly reduced to matrix completion, recognized as one of the widely referenced problems in the recommender systems and user response predictions [1] with numerous collaborative filtering and matrix factorization approaches [3, 21, 24, 27, 30, 42] and extended to deep learning-based approaches [44, 52, 59]. *Deep matrix factorization (DMF)* [57] directly uses the input matrix by feeding this information through multilayer perceptrons (MLPs). Because of its direct reliance on the input matrix, the model is forced to replace

the missing values (NaNs) with an arbitrary value which functions as an undesired bias during training. An extension to variational autoencoders (VAE) [19], the *partial-VAE model (PVAE)* [25] is an encoder-decoder-based approach for predicting the missing values, and VAEM [26] is an extension of this work which accounts for heterogeneous feature types. In contrast to these models, CoRGi leverages additional feature information such as node content in a user-item-specific manner. Each portion of content is used with different weights for different users.

Over the past years, there have been attempts to model graphs with entities as nodes and their relationships as edges [12, 36, 47]. Kipf and Welling [20] proposed *graph convolutional networks (GCNs)*, a convolutional neural network-based model that learns latent representations of nodes, amongst other deep neural network-based approaches [5, 7, 9, 22, 32, 43]. *GraphSAGE* by Hamilton et al. [13] extends GCN by allowing the model to be trained on some part of the graph, making the model to be used in inductive settings. As a GNN model that is designed for recommender systems, *graph convolutional matrix completion (GC-MC)* [2] is a variant of GCN that explicitly uses edge labels as inputs to model messages. Compared to other approaches, GC-MC employs a single-layer message-passing scheme, and each label is endowed with a separate message passing channel. *GRAPE* [60] employs edge embeddings on GCN and adopts edge dropouts that are applied throughout all message-passing layers. LightGCN [14] designs a GCN framework that simplifies or omits constructions that are not beneficial for recommendations, such as feature transformation and nonlinear activation, and puts more emphasis on the neighborhood aggregation. Compared to the previously proposed GNN models in recommender systems, CoRGi leverages the rich content information of nodes to model a target domain projected to graphs. Compared to other GNN models that exploit the content information of nodes, e.g., PinSAGE by Ying et al. [58], CoRGi employs the attention mechanism between a node and contents within another node. Therefore, CoRGi computes user-item-specific attention probabilities of a word (word tokens), which in turn are used to update the edge embeddings. Wu et al. [56] surveys the nascent literature on GNNs in recommender systems.

In natural language processing, the self-attention mechanism is used to relate word or word tokens at different positions of a given sequence and to create a latent representation of the word and sequence [23, 33, 34, 50]. Many GNN models [10, 15, 17, 62], with *graph attention networks (GATs)* [51] being a popular example, use the attention mechanism to allow the target nodes to distinguish the weights of multiple messages from the source nodes for aggregation. We note that our approach is orthogonal to the GNN models with attention layers; although CoRGi uses attention, it is over the content within each node instead of the neighbors of each node (in GATs). Thus, in future work CoRGi-like mechanisms can be embedded to GAT-like GNNs in replacement of GCNs.

4 EVALUATION

Model configuration. We employ the $L = 3$ CoRGi with node-embedding and edge-embedding cardinalities for all layers, $C^{(l,h)}$ and $C^{(l,e)} \forall l \in \{1, \ldots, L\}$ set to 64, and the prediction MLP has a single hidden layer of size 256. We initialize all node embeddings with random values, and assign the train label values to initialize the edge embeddings. We use mean pooling for aggregation. For the non-linear activation, we use LeakyReLU for attention coefficient computation with negative slope set to 0.2, as suggested by Veličković et al. [51] and ReLU [31] for the rest.

Training configuration. For all experiments, we train CoRGi with backpropagation [41] using the Adam optimizer [18] and a learning rate set to 0.001. We employ early stopping on validation loss during training, with train, test, and validation sets split in 8:1:1 ratio. We use binary cross entropy loss (BCE) for predicting binary values and mean squared error (MSE) for the ordinal datasets. We apply dropout [45] on the message passing layers and on prediction MLPs, as well as on edges [60], with dropout rates chosen from {0.1, 0.3, 0.5, 0.7} with respect to the validation set

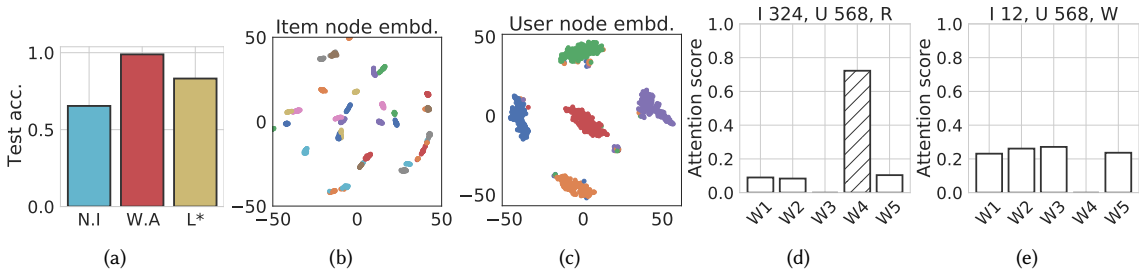


Fig. 2. Synthetic experiment results. (a): Test accuracies when using item content information for node initialization (N.I.), for computing content attentions (W.A.), and when the the edge labels are given (L∗). (b), (c): t-SNE plots of learned item and user node embeddings using CoRGi. Each dot represents a single node and is colored by its word distribution (item) and word-attentiveness (user). (d), (e): Computed word-attention scores of a user with correct and incorrect answers.

performance. For the comparison models, the parameter settings done in the following manner: 1) When the settings of the comparison models overlap with that of CoRGi, e.g., the number of message passing layers or the learning rate, we used the same configurations as CoRGi. 2) For the parameter settings that are unique to the comparison model, we followed the setting that is disclosed in the paper, with the following exceptions. 3) When the setting disclosed in the original paper is not applicable in for the datasets used for our experiment or our training environment, we selected settings that yielded the best performance within such restrictions.

4.1 Synthetic experiments

First, we construct a synthetic dataset to validate our model design. We create a bipartite graph with item and user nodes. Each item-node is associated with a number of "words" as its content. We use a "vocabulary" of 5 words and each item node contain each of the 5 words with 50% probability. Each user-node is assigned a single *focus word* that indicates the word the user "likes". Finally, the value of an edge between a user and an item is deterministically set to 1 if the item contains the user's "focus word", and 0 otherwise. Throughout experiments, the word-content of items is provided as an input to CoRGi, but the user focus word is latent. In this synthetic dataset, we are interested in seeing whether CoRGi discerns the correct edge labels between the users and the items. Furthermore, by looking at the attention scores, we analyze the capability of CoRGi to learn to focus on the user's focus word within each item node, if it is present.

Figure 2a shows the test accuracy of a GCN [20] with item node initialization using word vectors (left, blue bar), CoRGi (middle, red bar), and GCN with edge embeddings initialized with edge labels (right, yellow bar). Unlike the last model, the first two models do not use the ground truth edge labels during training. CoRGi is the only method that achieves near perfect test accuracy (> 0.99).

Figure 2b and 2c illustrate the t-SNE [48] visualization of computed user-node and item-node embeddings for CoRGi. Item and user nodes are colored by their associated content-word distributions and word-attentiveness, showing that the node embeddings can discriminate nodes by their attributes. Figure 2d and 2e display the computed attention scores between user-item pairs for two sample pairs. When the word that the user "likes" is included in the item's associated words, CoRGi correctly targets that word by assigning high attention score (Figure 2d). When word that user "likes" is absent, the attention distribution over the content-words of an item-node becomes much more uniform.

Table 2. Dataset statistics. ($|\mathcal{D}|$: Vocabulary size, $|\overline{\mathbf{D}}|$: average number of words per item, Density: graph density, #L: number of labels.)

	Nodes		Edges			Contents		Density	# L
	# Users	#Items	# Edges	/ user	/ item	$\lvert\mathcal{D}\rvert$	$\lvert\overline{\mathbf{D}}\rvert$		
Synthetic	1,000	1,000	100,000	100	100	5	2.5	0.1	2
Eedi	35,073	22,931	991,740	28	43	21,072	20.02	0.001	2
Goodreads	2,243	2,452	114,839	51	47	35,111	132.32	0.021	5

4.2 Evaluation on real-world data

Datasets. We evaluate CoRGi on two real-world datasets that record different user-item interactions (Table 2). The Goodreads dataset [16] from the Goodreads website contains users and books. The content of each book-node is its natural language description. The dataset includes a 1 to 5 integer rating between some books and users. The Eedi dataset [54] contains anonymized student and question identities with the student responses to some questions. The content of each question-node is the question text. Edge labels are binary: one (zero) for correct (incorrect) answers.

For both datasets, to encode the content within nodes in CoRGi, we use a pre-trained transformer encoder model [8] as $\mathbf{F}$. We use a *truncation threshold* T so that we ignore words that appear after T for any $\mathbf{D}_m$ with $n(m) > T$. The parameters of the GNN of CoRGi and the prediction multi-layer perceptron (MLP) (Equation 7) are learned jointly during training but we do *not* fine-tune the parameters of $\mathbf{F}$. We set $T = 64$ for both Goodreads and Eedi datasets.

We chose the Goodreads and Eedi datasets because they contains text information in sentences associated with each item. We use Goodreads dataset from Jannesar and Ghaderi [16]. We filtered out books whose descriptions are written in non-English languages, and removed duplicate books based on their titles. Originally, the ratings were text-based. We converted the ratings as follows: "Did not like it" to rating 1, "It was okay" to rating 2, "Liked it" to rating 3, "Really liked it" to rating 4, and "It was amazing" to rating 5.

We used Eedi dataset from Wang et al. [54]. The content of the text information is extracted using optical character recognition (OCR) from the raw question images, as no question text is available.

In order to train CoRGi and comparison models on a single GPU within our computation infrastructure, we took a subset of the Eedi dataset, taking student responses from between March 4^{th} and March 27^{th}.

Configurations for the baseline methods. We detail the configurations used specific to each baseline for recording the test performance. For GC-MC, we assign the separate message passing channels and their corresponding parameters for modeling different discrete edge labels. The number of layer is set to 1, and We do not use the weight sharing method. For the accumulation method, we use concatenation. For GraphSAGE, we use the neighbor sampling size of 32 throughout all message passing layers. For GRAPE, we do not use the one-hot node initializations for both Eedi and Goodreads, because of the large number of item nodes leading to GPU memory errors. Instead, we use random initialization just like all GNN model configurations in our experiment. For GAT, we use a single self-attention head.

Computing infrastructure. Each experiment was run on a single GPU, which was either an NVIDIA Tesla K80 or an NVIDIA Tesla V100. All experiments were scheduled and performed in Azure Machine Learning.

Baselines. We compare CoRGi with 7 widely used missing value imputation models, discussed in section 3. Deep matrix factorization (DMF) [57] and Partial variational autoencoder (PVAE) [25] are non-GNN matrix completion models. Graph convolutional network (GCN) [20], GraphSAGE [13], and Graph attention network [51] are GNN-based models not specifically designed to be operationalized in recommender systems. We compare CoRGi with these models by using a read-out MLP that accepts the concatenation of user and item node embeddings and makes a prediction for the pair. Finally, we also compare to graph convolutional matrix completion (GC-MC) [2] and GRAPE [60] that are GNN-based models for matrix completion.

None of the previous models consider content. Thus, we additionally consider 6 GCN-based models that use nodes content in a variety of ways:

(1) *GCN with WordNodes.* For each word in the content of all items, we create a new "word node". Each word node is then connected to a node $v_m \in \mathcal{V}_M$ if it is contained in the item's content. This baseline allows message passing word-specific information but the order of the words within each item is ignored. We retrieve words by stemming [38, 49], filtering non-alphanumeric words, and removing words with frequency of less than 2.
(2) *GCN Node Init: BoW* is a standard GCN with the initial node embeddings initialized with a multi-hot bag-of-words of the content. Words are tokenized and stemmed as above.
(3) *GCN Node Init: NeuralBoW* uses a pre-trained word2vec model [28, 29], implemented in Gensim [39]. Words are encoded in 300-dimensional vector representations and average pooling is used for node initialization.
(4) *GCN Node Init: BERT CLS* uses pre-trained, cased BERT base model [8], impemented in HuggingFace [55]. Words are encoded in 768-dimensional vector representations. We use the encoded representation of `[CLS]` to initalize the node embeddings.
(5) *GCN Node Init: BERT Avg* uses identical settings as GCN Node Init: BERT CLS, but instead of the `[CLS]` token, average pooling over all the output vector representations is used.
(6) *GCN Node Init: SBERT* uses the 768-dimensional vector representations encoding of the whole document with the SBERT model of Reimers et al. [40].

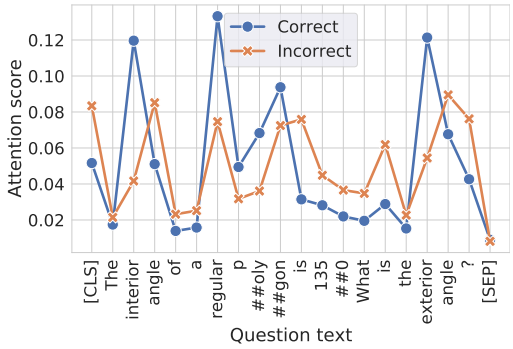


Fig. 3. Content attention distributions of students with correct and incorrect answers.

Table 4. Mean entropy of content attention distributions for reader-book pairs of different ratings (Goodreads) and student-question pairs of different answers (Eedi), with one standard error. Positive ratings (Goodreads) and correct answers (Eedi) have lower entropy values.

Goodreads		**Eedi**	
R > 3	R≤3	Correct	Incorrect
$2.39_{\pm 0.004}$	$2.58_{\pm 0.006}$	$2.64_{\pm 0.001}$	$2.72_{\pm 0.001}$

Missing value imputation. Table 3 compares the missing value imputation performances on test sets of the Goodreads and Eedi datasets. We report root mean square error (RMSE) for the Goodreads dataset and accuracy, area under the receiver operating characteristic (AUROC), and area under the precision-recall curve (AUPR) for Eedi. Overall, we observe improved performance when the node content information is used. Compared to the baseline GCN models

Table 3. Average test RMSE (Goodreads, lower is better) and test accuracy, AUROC, AUPR (Eedi, higher is better) results over 5 independent runs followed by one standard error. Best results are highlighted in bold, and the second-best results are underlined. $*$ and $**$ signify p-values less than 0.05 and 0.001 respectively from independent t-tests with the second-best results.

Model	Content	**Goodreads**	**Eedi**		
		RMSE	Accuracy	AUROC	AUPR
DMF	✗	$0.921_{\pm 0.001}$	$0.738_{\pm 0.002}$	$0.653_{\pm 0.003}$	$0.828_{\pm 0.002}$
PVAE	✗	$0.894_{\pm 0.001}$	$0.746_{\pm 0.001}$	$0.682_{\pm 0.000}$	$0.834_{\pm 0.000}$
GC-MC	✗	$0.916_{\pm 0.002}$	$0.735_{\pm 0.001}$	$0.672_{\pm 0.002}$	$0.819_{\pm 0.001}$
GCN	✗	$0.893_{\pm 0.001}$	$0.746_{\pm 0.002}$	$0.680_{\pm 0.001}$	$0.830_{\pm 0.001}$
GraphSAGE	✗	$0.898_{\pm 0.003}$	$0.742_{\pm 0.003}$	$0.665_{\pm 0.002}$	$0.838_{\pm 0.003}$
GRAPE	✗	$0.894_{\pm 0.001}$	$0.746_{\pm 0.001}$	$0.672_{\pm 0.001}$	$0.824_{\pm 0.001}$
GAT	✗	$0.893_{\pm 0.002}$	$0.745_{\pm 0.000}$	$0.684_{\pm 0.001}$	$0.832_{\pm 0.001}$
GCN with WordNodes	✓	$\underline{0.886}_{\pm 0.002}$	$0.751_{\pm 0.002}$	$0.710_{\pm 0.002}$	$0.839_{\pm 0.003}$
GCN Init: BoW	✓	$0.891_{\pm 0.001}$	$0.748_{\pm 0.001}$	$\underline{0.710}_{\pm 0.001}$	$0.836_{\pm 0.000}$
GCN Init: NeuralBoW	✓	$0.886_{\pm 0.001}$	$0.751_{\pm 0.000}$	$0.706_{\pm 0.001}$	$0.848_{\pm 0.001}$
GCN Init: BERT CLS	✓	$0.889_{\pm 0.001}$	$0.748_{\pm 0.001}$	$0.706_{\pm 0.001}$	$0.841_{\pm 0.001}$
GCN Init: BERT Avg.	✓	$0.887_{\pm 0.001}$	$0.750_{\pm 0.001}$	$0.708_{\pm 0.001}$	$0.848_{\pm 0.001}$
GCN Init: SBERT	✓	$0.890_{\pm 0.000}$	$\underline{0.752}_{\pm 0.002}$	$0.708_{\pm 0.002}$	$\underline{0.848}_{\pm 0.001}$
CoRGi: Concat	✓	$\mathbf{0.879}^{**}_{\pm 0.000}$	$\mathbf{0.756}^{*}_{\pm 0.001}$	$\mathbf{0.715}^{**}_{\pm 0.001}$	$\mathbf{0.874}^{**}_{\pm 0.000}$
CoRGi: Dot-product	✓	$\mathbf{0.879}^{**}_{\pm 0.000}$	$\mathbf{0.757}^{**}_{\pm 0.001}$	$\mathbf{0.717}^{**}_{\pm 0.001}$	$\mathbf{0.874}^{**}_{\pm 0.001}$

with content, both constructions of CoRGi with concatenation and dot-product outperforms on both datasets with statistical significance. Figure 3 shows the content attention distributions of user (student) - item (question) pairs in Eedi dataset for a particular question. The blue circle line shows the average attention scores of students who got the question right, and the orange cross line shows that of students with incorrect answers. We observe *user-item-specific* attention scores assigned; the student with right answer has high attention scores for word tokens `interior`, `regular`, and `exterior`, while the student with wrong answer attends more to word tokens `angle`, `angle`, and `[cls]`. This is in contrast to baseline models that use contents exploit the content information of items in a way that does not explicitly distinguish users during the message passing. We also observe (Table 4) that the distribution of the attention to content for correct answers (Eedi) and highly-scored books (Goodreads) has lower entropy compared to incorrect answers or low book ratings. This matches our intuition that knowing the answer to a question or liking a book is important.

Table 5. Mean ± one standard error of rating prediction (Goodreads, test RMSE) and response prediction (Eedi, test accuracy) results of all users (All), users with node degree greater than 10 ($D > 10$), and users with degrees less than or equal to 10 ($D \leq 10$). $*$ and $**$ signify p-values less than 0.05 and 0.001 respectively from paired t-tests.

	Goodreads			**Eedi**		
	All	D > 10	D ≤ 10	All	D > 10	D ≤ 10
GCN	$0.792_{\pm 0.007}$	$0.796_{\pm 0.008}$	$0.752_{\pm 0.03}$	$0.746_{\pm 0.002}$	$0.751_{\pm 0.002}$	$0.720_{\pm 0.008}$
GCN N.I.: SBERT	$0.786_{\pm 0.006}$	$0.789_{\pm 0.007}$	$0.728_{\pm 0.03}$	$0.753_{\pm 0.003}$	$0.757_{\pm 0.002}$	$0.727_{\pm 0.008}$
CoRGi (DP)	$\mathbf{0.781}^{*}_{\pm 0.006}$	$\mathbf{0.786}_{\pm 0.007}$	$\mathbf{0.685}^{**}_{\pm 0.03}$	$\mathbf{0.758}^{*}_{\pm 0.002}$	$\mathbf{0.760}^{*}_{\pm 0.003}$	$\mathbf{0.750}^{**}_{\pm 0.010}$

Sparsity analysis. Table 5 shows rating and response prediction performance on Goodreads and Eedi vs. the user node degree (D), i.e., the number of questions answered or books rated. When averaged over all users, CoRGi outperforms the baselines with $p < 0.05$ from paired t-tests with respect to GCN Node Init: SBERT (First column for each dataset). On both datasets, the predictive performance between CoRGi and the comparison models is more comparable for users that have interacted with more items ($D > 10$: second column for each dataset), although CoRGi still outperforms them. On the contrary, the difference in performance between CoRGi and the comparison models becomes more significant for users connected with less than or equal to 10 items ($D \leq 10$: third column for each dataset), showing the effectiveness of CoRGi on cold-start problems. Figure 4 shows test accuracy on Eedi with varying degrees, showing an increasing gap between CoRGi and baselines with smaller Ds.

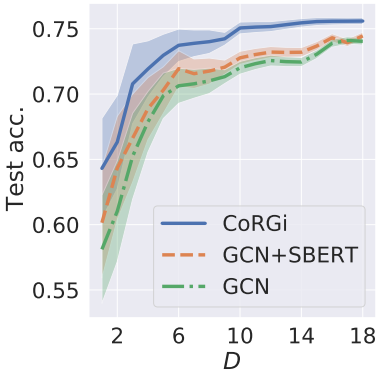
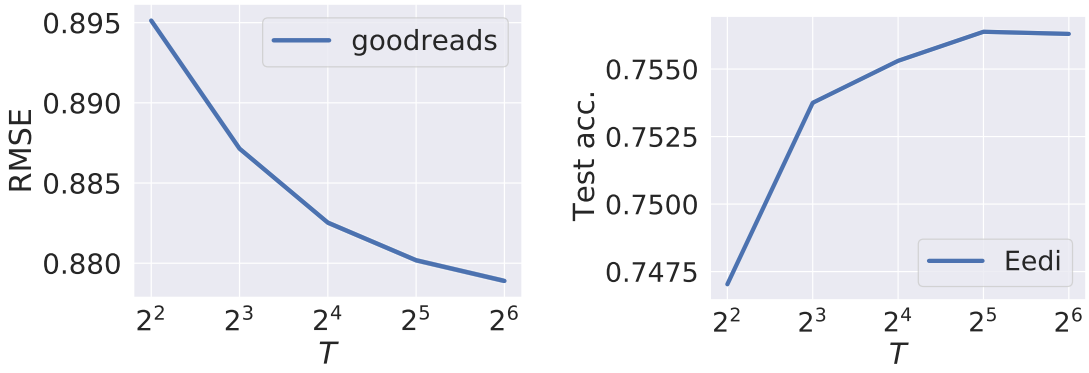


Fig. 4. Test acc. vs. user node deg. for Eedi.



(a) (b)

Fig. 5. (a) Truncation size and test accuracy for the Goodreads dataset. Note semi-log-x and starting point for y-axis. (b) Truncation size and RMSE for the Eedi dataset. Note semi-log-x and starting point for y-axis.

Truncation threshold and test performance. We test the affect of the truncation threshold T on the test performance. In sequential content encoders $\mathbf{F}$ such as the transformer encodes $\mathbf{D_m}$, i.e., the contents of an item node v_m of size $n(m)$ into a set of content vector representations $\mathbf{Z}_m$. During encoding, if $n(m)$ is greater than the truncation threshold T, we only the first T words only, i.e., $|\mathbf{Z}_m| = \min(n(m), T)$. Setting T to a high value enables CoRGi to fully exploit the content information, at a trade-off that makes the models slow to train with larger memory requirement.

Figures 5a and 5b show the test performance with respect to varying values of T on the Goodreads and Eedi datasets. For both datasets, increasing T results in higher test performance (T greater than 64 results in the memory error on our computing infrastructure). The average number of words on Eedi questions 20.02 (Table 2), and the test performance converges at $T = 32$. On the other hand, the average number of words on Goodreads book descriptions is 132.32, and we observe that the test performance does not fully converge at $T = 64$.

Performance comparison on caching trick. In subsection 2.2, we introduce a caching trick that allows the reduction in training time and memory requirement. Specifically, the caching trick is realized by creating a cache for $\mathbf{e}_{ij,\text{CA}}^{(l)}$ with zero initializations. We then update at the final layer L only, by computing $\mathbf{e}_{ij,\text{CA}}^{(L)}$ using Equation 4 and updating the cache for all $\mathbf{e}_{ij,\text{CA}}^{(l)}$ to the computed $\mathbf{e}_{ij,\text{CA}}^{(L)}$. Using the caching trick along with the neighbor sampling [13], the time complexity reduces from

$$O\left(|\mathcal{V}_U| \cdot C^{(l-1,h)} \cdot C^{(l,e)} + |\mathcal{V}_M| \cdot T \cdot D \cdot C^{(l,e)} + |\mathcal{E}| \cdot T \cdot C^{(l,e)}\right)$$

to

$$O\left(|\mathcal{N}(\mathcal{V}'_M)| \cdot C^{(l-1,h)} \cdot C^{(l,e)} + |\mathcal{V}'_M| \cdot T \cdot D \cdot C^{(l,e)} + |\mathcal{E}'| \cdot T \cdot C^{(l,e)}\right),$$

where we sample a subset of nodes $\mathcal{V}' = \left\{\mathcal{V}'_U \cup \mathcal{V}'_M\right\}$ for the neighbor sampling and only update $\mathbf{e}_{ij,\text{CA}}$ whose target node v_j is in $\mathcal{V}'_M$ and source node v_i is in $\mathcal{N}(\mathcal{V}'_M)$.

Table 6. Performance comparison with and without the caching trick on Goodreads dataset.

	RMSE	Training time / iteration (sec.)
With caching	$0.879_{\pm 0.000}$	10.41
Without caching	$0.879_{\pm 0.000}$	41.86

Table 6 compares the predictive performance of CoRGi with and without the caching trick on Goodreads dataset in terms of RMSE and wall-clock training time. The predictive performance comparison on Eedi dataset was not feasible due to excessive memory requirement in the absence of the caching trick. In the absense of the caching trick, the predictive performance remains the same, but the training time is increased more than 4 times per iteration.

Comparison on different combination methods. In section 2 we introduce two ways to augment the computed content-attention (CA) edge embeddings $\mathbf{e}_{ij,\text{CA}}^{(l)}$ to edge embeddings $\mathbf{e}_{ij}^{(l)}$ between nodes v_i and v_j at l^{th} layer. The first method is to first update the edge embeddings with the content attention ($\mathbf{e}_{ij}^{(l)\prime}$) and use the element-wise addition. The second method is to concatenate with the input edge feature $\mathbf{e}_{ij}^{(0)}$. We compare the predictive performances of these methods for the Goodreads and Eedi datasets. In tables 7 and 8, element-wise addition yields better predictive performance than concatenation for on both datasets with varying methods of attention computation: concat and dot-product.

Table 7. Comparison of different combination methods for updating the edge embedding $\mathbf{e}_{ij}^{(l)}$ on the Goodreads dataset. We report RMSE (lower the better).

Combination method	CoRGi :Concat	CoRGi :Dot-product
$\mathbf{e}_{ij}^{(l)\prime} + \mathbf{e}_{ij,\text{CA}}^{(l)}$	$0.879_{\pm 0.000}$	$0.879_{\pm 0.000}$
$\textsc{Concat}(\mathbf{e}_{ij}^{(0)}, \mathbf{e}_{ij,\text{CA}}^{(l)})$	$0.886_{\pm 0.000}$	$0.884_{\pm 0.001}$

Table 8. Comparison of different combination methods for updating the edge embedding $\mathbf{e}_{ij}^{(l)}$ on the Eedi dataset. We report test accuracy (higher the better).

Combination method	CoRGi :Concat	CoRGi :Dot-product
$\mathbf{e}_{ij}^{(l)\prime} + \mathbf{e}_{ij,\text{CA}}^{(l)}$	$0.756_{\pm 0.001}$	$0.757_{\pm 0.001}$
$\textsc{Concat}(\mathbf{e}_{ij}^{(0)}, \mathbf{e}_{ij,\text{CA}}^{(l)})$	$0.752_{\pm 0.001}$	$0.752_{\pm 0.001}$

5 CONCLUSION

In this work, we presented CoRGi, a GNN based framework that tightly integrates content within the nodes using personalized attention. Using content — such as text — present in the modelled graph allows us to capture rich information within the target domain while maintaining the structured form of the data. This is especially evident in sparse regions of the graphs showing in different real-world edge value prediction tasks. While CoRGi presented one effective way to integrate message-passing in GNNs with an attention mechanism over content, future work needs to further investigate how additional modalities beyond text can be captured in content-rich graphs in a broader range of applications beyond edge value prediction tasks.

As for the future work, it might be worthwhile to investigate text-augmented graphs where texts are associated not only for the item nodes but also for the user nodes as well. CoRGi can naturally operate on this setting because the personalized attention mechanism considers directionality and the message floating from user to item can be different from that floating from item to user. Furthermore, another potential future direction is to work on content metadata outside of texts: for example, pixel-based images.

REFERENCES

[1] James Bennett, Stan Lanning, et al. 2007. The Netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. 35.

[2] Rianne van den Berg, Thomas Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In *KDD Deep Learning Day Workshop*.

[3] Daniel Billsus, Michael Pazzani, et al. 1998. Learning collaborative information filters. In *ICML*.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL* 5 (2017), 135–146.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[9] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*.

[10] Hongyang Gao and Shuiwang Ji. 2019. Graph representation learning via hard and channel-wise attention networks. In *KDD*.

[11] Justin Gilmer, Samuel Schoenholz, Patrick Riley, Oriol Vinyals, and George Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.

[13] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.

[15] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2020. Measuring and improving the use of graph information in graph neural networks. In *ICLR*.

[16] Bahram Jannesar and Soursh Ghaderi. 2020. Goodreads book dataset. https://github.com/BahramJannesar/GoodreadsBookDataset.

[17] Dongkwan Kim and Alice Oh. 2021. How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision. In *ICLR*.

[18] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

[19] Diederik Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *ICLR*.

[20] Thomas Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[22] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *ICLR*.

[23] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

[24] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[25] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. 2019. EDDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE. In *ICML*.

[26] Chao Ma, Sebastian Tschiatschek, Richard Turner, José Miguel Hernández-Lobato, and Cheng Zhang. 2020. VAEM: a Deep Generative Model for Heterogeneous Mixed Type Data. In *NeurIPS*.

[27] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

[29] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *NAACL*.

[30] Andriy Mnih and Russ Salakhutdinov. 2007. Probabilistic matrix factorization. In *NeurIPS*.

[31] Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *ICML*.

[32] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*.

[33] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.

[34] Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *ICLR*.

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*.

[37] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*.

[38] Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

[39] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en.

[40] Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.

[41] David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.

[43] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.

[44] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.

[46] Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

[47] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*.

[48] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).

[49] Cornelis J Van Rijsbergen, Stephen Edward Robertson, and Martin F Porter. 1980. *New models in probabilistic information retrieval*. Vol. 5587. British Library Research and Development Department London.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*.

[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[52] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.

[53] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.

[54] Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Yordan Zaykov, José Miguel Hernández-Lobato, Richard E Turner, Richard G Baraniuk, Craig Barton, Simon Peyton Jones, et al. 2020. Diagnostic questions: The NeurIPS 2020 education challenge. *arXiv preprint arXiv:2007.12061* (2020).

[55] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[56] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260* (2020).

[57] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*.

[58] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*.

[59] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *ICML*.

[60] Jiaxuan You, Xiaobai Ma, Daisy Yi Ding, Mykel Kochenderfer, and Jure Leskovec. 2020. Handling missing data with graph representation learning. In *NeurIPS*.

[61] Yongjian You, Weijia Jia, Tianyi Liu, and Wenmian Yang. 2019. Improving abstractive document summarization with salient information modeling. In *ACL*.

[62] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294* (2018).

[63] Muhan Zhang and Yixin Chen. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In *ICLR*.