Graph Attention Networks over Edge Content-Based Channels

Lu Lin, Hongning Wang
Department of Computer Science
University of Virginia
Charlottesville, VA 22904, USA
{ll5fy,hw5x}@virginia.edu

ABSTRACT

Edges play a crucial role in passing information on a graph, especially when they carry textual content reflecting semantics behind how nodes are linked and interacting with each other. In this paper, we propose a channel-aware attention mechanism enabled by edge text content when aggregating information from neighboring nodes; and we realize this mechanism in a graph autoencoder framework. Edge text content is encoded as low-dimensional mixtures of latent topics, which serve as semantic channels for topic-level information passing on edges. We embed nodes and topics in the same latent space to capture their mutual dependency when decoding the structural and textual information on graph. We evaluated the proposed model on Yelp user-item bipartite graph and StackOverflow user-user interaction graph. The proposed model outperformed a set of baselines on link prediction and content prediction tasks. Qualitative evaluations also demonstrated the descriptive power of the learnt node embeddings, showing its potential as an interpretable representation of graphs.

CCS CONCEPTS

• **Computing methodologies → Learning latent representations**; **Latent variable models**; *Neural networks*; *Latent Dirichlet allocation.*

KEYWORDS

Graph neural networks; representation learning; topic modeling; variational auto-encoder

ACM Reference Format:
Lu Lin, Hongning Wang. 2020. Graph Attention Networks over Edge Content-Based Channels. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403233

1 INTRODUCTION

Graphs are ubiquitous and exist in a wide variety of forms in real world, including social networks [23], user-item bipartite connections [3], citation networks [30], and many more [2, 24]. As one of the most general forms of data representation, tremendous research attention has been constantly devoted in analyzing, modeling, and mining graphs [7, 31].

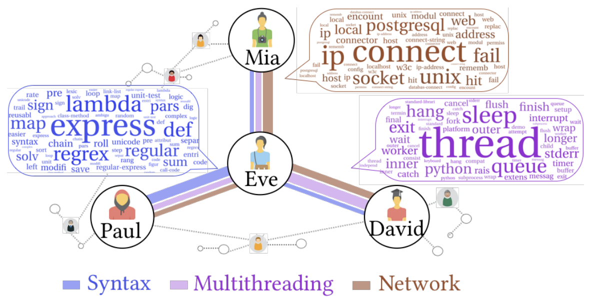


Figure 1: A motivating example of content-based edge channels across connected nodes. Color indicates topics, and width of colored areas on edge reflects the topic proportion on each connection.

A graph utilizes edges to encode relational knowledge about interacting nodes. Edges, instead of being simply binary indicators of connectivity, are often in possession of rich and multi-modal information. For example in *multiplex* and *attributed* graphs [8, 34], a set of attributes are associated with edges to represent *what* types of proximity exist between nodes. Unlike attributes with explicitly predefined meanings, unstructured text content on edges carries latent but informative semantics that manifest *how* and *why* nodes are related. Text content tied with edges is also pervasive in real-world graphs. This urges us to leverage such rich semantic content to unleash the potential of reasoning and explaining the interactions between nodes, e.g., an interpretable graph representation.

To illustrate this new perspective for graph modeling, Figure 1 shows the motivating example of our work in this paper, which is extracted from a set of real users' interactions on Stack Overflow[1]. We visualize the text content that users post to each other by word clouds. The content suggests that all four users' interactions focus on different aspects of "*programming*", but their interactions clearly take distinct concentrations. For instance, the interactions between Eve and Mia are mostly about "*socket, IP, host*", reflecting their shared interest on the topic of "*network programming*", while Eve and David are more likely to interact on "*multithreading*". Without such fine-grained signals about the nodes' relations, it is hard for a statistical model to accurately characterize the nodes solely from the graph structure, not mentioning to infer the dependency among them (e.g., how they are related).

In spite of the ubiquity and informativeness of textual edge content, little effort has been devoted to utilizing such semantic information in modeling graphs. Current mainstream effort focuses

[1]Stack Overflow: http://stackoverflow.com

on modeling graph structures, e.g., adjacency matrix or Laplacian matrix [16, 25]. Such solutions are designed to assign similar representations to nodes with a similar neighborhood structure, but cannot further differentiate the nuance among such nodes. As shown in Figure 1, classical graph embedding methods tend to assign Mia and Dave similar embedding vectors, because they share connections with Eve. But the edge content suggests otherwise, as they interact with Eve for different reasons. Some recent efforts address this issue by manually crafted edge features that generate edge-specific transformation matrices when aggregating node embeddings from neighbors [9, 14]. Yang et al. [32] also consider multi-modal edges by assuming a distribution of multiple relations underlying each edge. But such solutions separate the modeling of edge text content from the modeling of graph structure, which creates a gap between the efforts on these two related modeling tasks.

Modeling edge text content imposes several new challenges, and thus can never be a straightforward extension of existing graph embedding solutions. First, compared to well-defined edge attributes or feature vectors [8], text content is highly unstructured, high-dimensional, and sometimes noisy. This poses difficulties to distill meaningful and low-dimensional information to reason about the associated node connections [5]. Second, text content is often a mixture of hidden semantics. For example, one piece of text might cover multiple correlated themes or topics, which is in a sharp contrast to edge attributes that are defined to be independent and mutually exclusive [32]. Third, text content on edges is mutually dependent on graph structure and node properties. For example in social networks, individuals are more likely to form ties with others sharing similar interests (known as homophily [22]); but once connected, they tend to generate content regarding the shared interest (known as social influence [10]). All these challenges require graph modeling to be performed hand in hand with content modeling, fusing graph structural dependency with edge content semantics.

In this work, we propose a fresh principle for graph embedding: nodes that are structurally and *semantically* similar should be close in the embedding space. We measure semantics underlying text content by *topics*, each of which is modeled as a probability distribution over a fixed vocabulary [5]. Edge content can then be represented as a topic distribution that characterizes the semantic relatedness between two interacting nodes. This leads to a fine-grained information aggregation mechanism on top of graph convolutions [20]: We decompose each edge into multiple *channels*, where each channel corresponds to a particular topic and the associated topic proportion indicates channel bandwidth for message passing. As a result, nodes will be embedded closer if they are more semantically related as suggested by the edge content about their interactions.

We realize the edge content based channel mechanism in a unified graph autoencoder framework, namely Channel-aware Graph Attention Network (CGAT). In this framework, information is aggregated by jointly attending neighboring nodes guided by both node embeddings and topical channels learned from edge text content. For each convolution layer in the encoder, we embed nodes and topics to the same latent space such that their mutual dependency is captured for a decoder to reconstruct the structural and textual information. The affinity between nodes is characterized by their proximity in the embedding space, which is used to reconstruct the connection structure. And the connected nodes are projected onto the topic space to measure affinity between topics and nodes, which is then used to construct the prior distribution of topics on the edge to regularize the reconstruction of edge text content. We infer the distribution of topics on the graph edges via a variational autoencoder, which provides an efficient and flexible solution for posterior inference in our model. Extensive experiments on two real-world large-scale graphs constructed from Yelp and StackOverflow demonstrate the expressive and predictive power of the proposed CGAT framework.

2 RELATED WORK

Recent years have witnessed a surge of interest in generalizing convolution operations in neural networks to modeling graphs, referred to as graph convolutional networks (GCNs). Early work define *spectral* convolution in the Fourier domain by computing the eigen decomposition of graph Laplacian [6]. Subsequent extensions and approximations [11, 20] are proposed to make the spectral filters spatially localized. This type of approaches have attracted tremendous attention due to their strong performance in a variety of tasks, such as node classification [4] and link prediction [28]. A major limitation of these traditional graph convolutional methods is that the filters are learned on the entire graph Laplacian, which lacks feasibility and scalability when graph is changed or includes more information. *Non-spectral* approaches [17], on the other hand, define convolutions directly on graph to aggregate information locally from neighbors, and maintain the weight sharing property of CNNs. Such solutions have yielded impressive performance across several large-scale inductive benchmarks [33].

The representability of GCNs largely depends on how states of nodes are transformed during message passing [12], and side information on nodes and edges is introduced to enable a variety of transformations. Node feature is incorporated in [1, 30] to reweigh the transformed neighbor states via a self-attention mechanism. Node textual information is used to capture the semantic correlation between nodes via a mutual attention mechanism [29]. Edge types are considered in multi-relational graphs, where each edge is associated with a particular type of relation, to generate type-specific transformation [8, 26, 32]. Multiple modalities of edges are modeled in [32], which assumes a distribution of multiple relations underlying each edge. To handle general edge features, Gong and Cheng [14] directly incorporate edge feature vectors to extend the attention mechanism first proposed in [30] by attending neighbors for each type of features. Gilmer et al. [13] introduce an edge network which takes the feature vectors of edges as input and outputs matrices to transform neighboring nodes' embeddings. As a follow up, Chen et al. [9] maximize the mutual information between the input feature matrix and the output transformation matrices to preserve edge features.

In these existing studies, incorporating node and edge content has been demonstrated effective for state transformation in information aggregation. However, content information considered on edges is mostly well-structured features, such as attributes and types. Textual content on edges carrying rich semantic information about what triggers the connection between the nodes, is surprisingly underexplored. In contrast, our work extracts low-dimensional semantics from edge text and enables a channel-aware information aggregation mechanism to exploit such signals.

3 METHODOLOGY

To guide information aggregation among nodes by the hidden semantics underlying edge text content, we propose a unified autoencoder framework, named Channel-aware Graph Attention Network (CGAT). In this section, we first present the key building block layer which decomposes edge as a group of topical channels, and propagates information according to the channels. Then we explain how to infer the latent channels from edge text content via a variational autoencoder. As a whole, these different components are joined by a reconstruction loss defined on the graph structure and edge content and estimated in an end-to-end fashion.

3.1 Channel-aware Attention Layer

In Figure 1, we illustrated our key insight that edge text content serving as a form of local context can differentiate connections semantically. It injects important edge-specific supervision when transforming and aggregating neighboring states. Following this insight, we propose to profile every edge on graph as a mixture of semantic-driven channels, which guide a fine-grain information aggregation through a channel-aware attention layer.

Formally, we assume K latent semantic channels and each edge is associated to these channels with different bandwidths. In the channel-aware attention layer, state from each neighbor is passed and attended with respect to each channel. The input of one layer has two parts: 1) **node states**, $\boldsymbol{h} = \{h_1, \ldots, h_{|V|}\}, h_i \in \mathbb{R}^m$, where $|V|$ is the number of nodes and m is the dimension of node embedding in the current layer; 2) **channel bandwidths** on all edges, $\Lambda = \{\lambda_{ij}\}_{e_{ij} \in E}, \lambda_{ij} \in \Delta^K$, where E is the edge set of the given graph with e_{ij} denoting the edge between node i and j. Δ^K is a probability simplex over K dimensions, i.e., $\forall k, \lambda_{ij}^k \geq 0$ and $\sum_{k=1}^K \lambda_{ij}^k = 1$. The layer outputs node states, $\boldsymbol{h}' = \{h'_1, \ldots, h'_{|V|}\}, h'_i \in \mathbb{R}^{m'}$. Note that multiple layers can be stacked, and the node states output in one layer will be used as input of the next layer. To unify our notations, we denote the raw input node features as $\boldsymbol{h}^{(0)}$, and the final embeddings output at the last layer L as $\boldsymbol{u} \equiv \boldsymbol{h}^{(L)}$.

To jointly attend neighbors and channels on each channel-aware attention layer, we propose the following aggregation operation:

$$h'_i = \sigma\Big(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \lambda_{ij}^k \alpha_{ij}^k W^k h_j\Big) \tag{1}$$

$\mathcal{N}_i$ is the neighborhood of node i and σ is the sigmoid function for non-linearity. There are two key components to realize our principle for graph embedding, i.e., structural and semantic similarity, which are explained as follows.

First, the aggregation is performed locally within each node's neighborhood, which preserves the structural information in node states. Second, λ_{ij}^k denotes the bandwidth of the k-th channel, which indicates the importance of this channel to the interaction between node i and j. $\{\lambda_{ij}^k\}_{k=1}^K$ are latent channel variables underlying the associated edge content on e_{ij}, and we will discuss how to infer them in the next section. Basically, λ_{ij}^k is expected to be larger if the edge content is more related to channel k to reveal semantic similarity between node i and j. Correspondingly, α_{ij}^k is the self-attention coefficient which measures the relatedness between node j's state to node i with respect to channel k. We follow GAT's

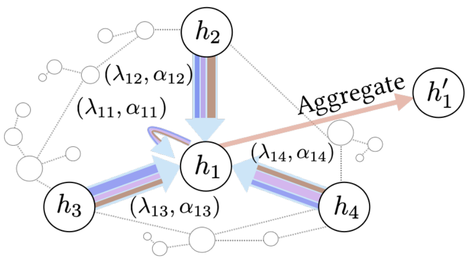

h_2
$(\lambda_{12}, \alpha_{12})$
$(\lambda_{11}, \alpha_{11})$
Aggregate
h'_1
$(\lambda_{14}, \alpha_{14})$
h_1
h_3
$(\lambda_{13}, \alpha_{13})$
h_4

Figure 2: Illustration of channel-aware attention layer. Each edge is decomposed by channels, denoted by different colors, with width of colored area indicating the importance of the channel between connected nodes.

multi-head attention scheme [30] to compute α_{ij}^k as follows,

$$\alpha_{ij}^k = \frac{\exp\left(\text{LeakyReLU}(a^{k\top} [W^k h_i \| W^k h_j])\right)}{\sum_{j' \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}(a^{k\top} [W^k h_i \| W^k h_{j'}])\right)} \tag{2}$$

where $\|$ is the concatenation operation over two vectors. For the k-th channel, $a^k \in \mathbb{R}^{2m'}$ is the weight vector of a single-layer feed-forward neural network, and $W^k \in \mathbb{R}^{m' \times m}$ is a shared linear transformation matrix over all edges in the graph. Each matrix W^k is explicitly coupled with a particular channel k. We handle the special case of self-loop by setting λ_{ii} to a uniform vector, if edge $e_{ii} \notin E$ or no text content is associated with edge e_{ii}.

Unlike GAT where different heads of attention are assumed to be independent and without any particular physical meaning, we explicitly marry attention with the underlying semantic content on edges. We use Figure 2 to depict the aggregation operation in channel-aware attention layer: when updating the state of node 1, we pass each neighbor's state through $K = 3$ semantic channels, subject to 1) channel bandwidth λ derived from edge content, and 2) attention score α based on node states. Thus, two nodes influence each other only when the semantics underlying their interaction is about a channel *and* they are also related under this channel.

The design of our channel-aware attention mechanism fulfills our proposed principle of graph embedding: nodes that are structurally and semantically similar should be close in the embedding space. Previous work that only considered graph structure ignored an important fact that nodes are inherently connected under multiple relational semantics [32]. As shown in Figure 1, users are connected based on different topical preferences reflected by the text content generated during their interactions. Essentially, information propagated from different neighbors is not semantic-equivalently received by the target node, which may result in structurally similar nodes differing greatly in a semantic sense. For example, Eve and Mia are connected for a different reason than that for Eve and David in Figure 1. The proposed channel-level aggregation explicitly imposes attention on different semantic channels on each edge, such that nodes will be embedded closer if they share similar semantics behind their interactions. Furthermore, the learned channels along with their inferred bandwidth can lead to fine-grain interpretability of each modeled connection, which will be shown as the case study in our later empirical study.

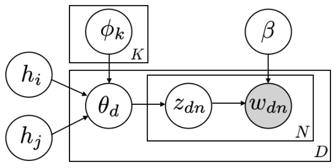


Figure 3: Graphical model representation of the text generation between node i and j, whose states are embedded as h_i and h_j. The upper plate indexed by K denotes topic embeddings for channels. The right plate indexed by D denotes the text documents on edge e_{ij}.

3.2 Channel Modeling

To infer latent semantics underlying unstructured edge textual content, we appeal to neural topic models, which are one of the most popular methods for learning unsupervised representations of text [5, 27]. As a result, the hidden semantics that form channels are defined as topics, and the bandwidths of semantic channels for each edge are modeled as a distribution over topics.

Without loss of generality, we assume edge between node i and node j is associated with a set of documents $\mathcal{D}_{ij} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|\mathcal{D}_{ij}|}\}$. Each document is denoted as a bag of words $\boldsymbol{w}_d = \{w_1, \ldots, w_N\}$, where each word is chosen from a vocabulary $\mathcal{V}$. Given K latent topics over the graph, we assume that each document $\boldsymbol{w}_d$ is generated by a mixture of latent topics specified by $\theta_d \in \Delta^K$. Since there may exist multiple documents between nodes i and j, averaging the topic proportions over all these documents produces the bandwidths on edge e_{ij} for the channel-aware attention layer:

$$\lambda_{ij} = \frac{1}{|\mathcal{D}_{ij}|} \sum\nolimits_{d \in \mathcal{D}_{ij}} \theta_d \tag{3}$$

Each of K topics is denoted as a probability distribution $\beta_k \in \Delta^{|\mathcal{V}|}$ over the vocabulary. We use $\beta = (\beta_1, \ldots, \beta_K)$ to represent the word distribution under all topics. Recall that the hidden state for node i is $h_i \in \mathbb{R}^m$ in our channel-aware attention layer, we embed each topic with a state vector $\phi_k \in \mathbb{R}^m$ in the same space, and use $\Phi \in \mathbb{R}^{K \times m}$ to denote the matrix of embeddings for all K topics [15].

For all documents associated with e_{ij}, we impose a shared prior topic distribution parameterized by $\Phi \cdot (h_i + h_j)$, which reflects the commonly shared topical preferences between the two nodes. Intuitively, $\Phi \cdot h_i$ projects node i into the topic space to measure its relatedness to each topic; and the prior on e_{ij} reflects the joint influence from node i and j. Other forms of prior distribution can also be introduced, if one has more explicit knowledge about the node and topic [21]. Based on this design of prior topic distribution, Figure 3 shows the generative process of edge text content between all pairs of nodes i and j in the graph, which can be described as follows:

- For each topic k:
  - Draw its topic state vector $\phi_k \sim \mathcal{N}(\mathbf{0}, \epsilon^2 I)$
- For each document d on edge between node i and j:
  - Draw its topic proportion vector (indicating shared preference between i and j) $\theta_d \sim \mathrm{Dir}\big(\mathrm{softmax}(\Phi \cdot (h_i + h_j))\big)$
  - For each word w_{dn}:
    - Draw its topic assignment $z_{dn} \sim \mathrm{Multi}\big(\mathrm{softmax}(\theta_d)\big)$
    - Draw its word $w_{dn} \sim \mathrm{Multi}(\beta_{z_{dn}})$

To simplify the notations, we will omit the subscript d and denote θ as the per-document topic distribution by default. The topic proportion can be obtained via posterior inference, $p(\theta, z|\boldsymbol{w}) = p(\theta, z, \boldsymbol{w})/p(\boldsymbol{w})$, which, however, is intractable caused by the coupling between latent variables θ and z in the marginal likelihood $p(\boldsymbol{w})$. To efficiently solve the problem, we appeal to the variational autoencoders (VAEs) [18] to approximate the posterior.

Specifically, we introduce a parametric variational model $q(\theta, z|\boldsymbol{w})$ that takes observed document as input to approximate the posterior, such that $q(\theta, z|\boldsymbol{w}) \approx p(\theta, z|\boldsymbol{w})$. The optimization problem of minimizing the KL divergence between these two distributions is equivalent to maximizing the evidence lower bound (ELBO):

$$\mathcal{L}(q, p) = \mathbb{E}_{\theta, z \sim q}\big[\log p(\boldsymbol{w}|\theta, z)\big] - D_{KL}\big(q(\theta, z|\boldsymbol{w}) \| p(\theta, z)\big) \tag{4}$$

where $q(\theta, z|\boldsymbol{w})$ and $p(\boldsymbol{w}|\theta, z)$ are referred to as *encoder* model and *decoder* model, respectively. To obtain unbiased gradients for the parameters in the encoder network, the "reparameterization trick" is widely employed [18, 19] when computing the expectation in Eq (4). To sidestep the difficulty of applying this technique on discrete variable z, we follow [27] to collapse z and only leave θ: $p(w_n|\theta) = \sum_{k=1}^{K} p(w_n|z = k)p(z = k|\theta) = (\theta^\top \beta)^{w_n}$, which eliminates the hidden variable z by directly drawing $w_n \sim \mathrm{Multi}(\theta^\top \beta)$. As a consequence, we simplify the ELBO with $q(\theta|\boldsymbol{w})$ as the encoder and $p(\boldsymbol{w}|\theta)$ as the decoder:

$$\mathcal{L}(q, p) = \mathbb{E}_{\theta \sim q}\big[\log p(\boldsymbol{w}|\theta)\big] - D_{KL}\big(q(\theta|\boldsymbol{w}) \| p(\theta)\big) \tag{5}$$

Given a document $\boldsymbol{w}$, the encoder network is set to a logistic normal distribution $q(\theta|\boldsymbol{w}) = \mathcal{LN}(\mu, \Sigma)$ with mean $\mu = \mathrm{MLP}_\mu(\boldsymbol{w})$ and variance $\Sigma = \mathrm{MLP}_\Sigma(\boldsymbol{w})$, where different multilayer perceptrons are used to compute μ and Σ based on document content. The decoder model is set to a multinomial distribution $p(\boldsymbol{w}|\theta) = \prod_{n=1}^{N} p(w_n|\theta) = \prod_{w \in \mathcal{V}} (\theta^\top \beta)^w$. Again, the prior is our imposed Dirichlet distribution $p(\theta) = \mathrm{Dir}\big(\mathrm{softmax}(\Phi(h_i + h_j))\big)$. We now illustrate the computation of Eq (5) in detail.

• **Compute** $\mathbb{E}_{\theta \sim q}\big[\log p(\boldsymbol{w}|\theta)\big]$. We utilize the reparameterization trick to transform the random variable $\theta \sim q(\theta|\boldsymbol{w})$ as a function of new variable ϵ: $\theta = \sigma(\mu + \varepsilon \Sigma^{1/2})$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}, I)$. The expectation is then rewritten over the reparameterized variable ε, which can be computed by sampling $\varepsilon \sim \mathcal{N}(\mathbf{0}, I)$:

$$\mathbb{E}_{\theta \sim q}\big[\log p(\boldsymbol{w}|\theta)\big] = \mathbb{E}_{\varepsilon \sim \mathcal{N}(\mathbf{0}, I)}\Big[\boldsymbol{w}^\top \log\big(\sigma(\mu + \varepsilon \Sigma^{1/2})^\top \sigma(\beta)\big)\Big] \tag{6}$$

• **Compute** $D_{KL}\big(q(\theta|\boldsymbol{w}) \| p(\theta)\big)$. Given it is yet unknown about how to develop an effective reparameterization function to handle the Dirichlet distribution, we use the Laplace approximation [27] to approximate the Dirichlet distribution by a logistic normal distribution $p(\theta) \approx \mathcal{LN}(\mu', \Sigma')$ with special forms of mean and diagonal variance defined as:

$$\mu'_k = \log \alpha_k - \frac{1}{K} \sum\nolimits_i \log \alpha_i, \ \Sigma'_{kk} = (1 - \frac{2}{K})\frac{1}{\alpha_k} + \frac{1}{K^2} \sum\nolimits_i \frac{1}{\alpha_i} \tag{7}$$

where $\alpha = \sigma\big(\Phi(h_i + h_j)\big)$ incorporates information from node states. Consequently, the KL divergence is between two logistic normals, which can be directly obtained by:

$$D_{KL}\big(q(\theta|\boldsymbol{w}) \| p(\theta)\big) = \frac{1}{2}\Big(\mathrm{Tr}(\Sigma'^{-1}\Sigma) + (\mu' - \mu)\Sigma'^{-1}(\mu' - \mu) + \log \frac{|\Sigma'|}{|\Sigma|}\Big) \tag{8}$$

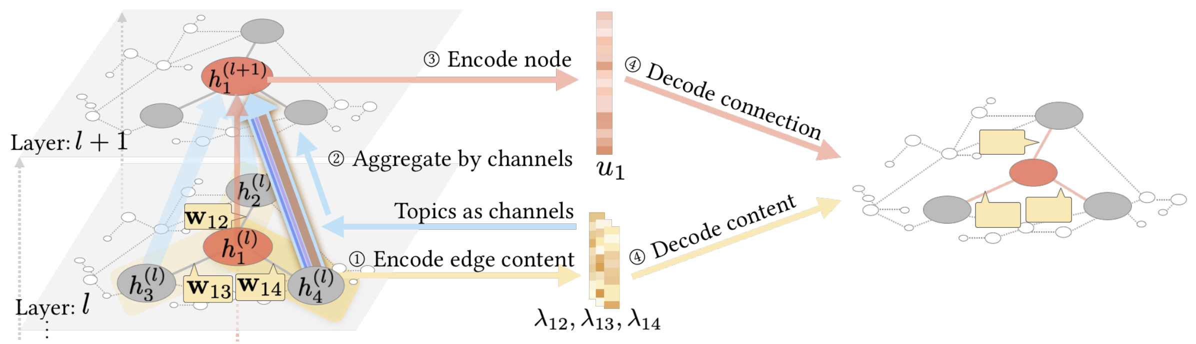


Figure 4: Illustration of the graph autoencoder architecture in CGAT. On the left side, CGAT uses the channel-aware attention layers to encode nodes and use the topic-driven channel model to encode edge content in the corresponding latent spaces. On the right side, CGAT decodes their embeddings to reconstruct the graph structure and edge text content.

Note that the objective function in Eq (5) is per-document. Each channel-aware attention layer will be associated with multiple documents, and the node embeddings are jointly optimized with respect to the topic posteriors. For efficiency concerns, at training time we randomly sample a batch of edges each time to compute this objective function.

3.3 Model Objective

To jointly learn node embeddings and topic embeddings while preserving the graph structural and edge textual information in an unsupervised setting, we construct the following loss function:

$$L = \sum_{e_{ij}\in E} \Big\{ -\log\left(\sigma(u_i^\top u_j)\right) - Q\cdot \mathbb{E}_{j_n\sim\overline{\mathcal{N}_i}} \log\left(\sigma(-u_i^\top u_{j_n})\right) \quad (9)$$
$$- \sum_{d\in\mathcal{D}_{ij}} \Big[\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)} \left[\boldsymbol{w}_d^\top \log\left(\sigma(\mu_d+\varepsilon\Sigma_d^{1/2})^\top\sigma(\beta)\right)\right]$$
$$- \frac{1}{2}\Big(\mathrm{Tr}(\Sigma_d'^{-1}\Sigma_d) + (\mu_d'-\mu_d)^\top \Sigma_d'^{-1}(\mu_d'-\mu_d) + \log\frac{|\Sigma_d'|}{|\Sigma_d|}\Big)\Big]\Big\}$$

where u_i is embedding of node i from the output layer, $\overline{\mathcal{N}_i}$ is node i's non-neighbor set from which Q negative examples are randomly sampled. The first term in Eq (9) represents the graph reconstruction error, which pushes structurally close nodes together and separates disconnected nodes in the embedding space. The rest terms, inherited from the ELBO for channel modeling, also carry evident intuitions: The second term is the edge text content reconstruction error, which encourages better topic representations from the observed texts; and the third term from KL divergence between the prior $p(\theta)$ and the approximated posterior $q(\theta|\boldsymbol{w})$ poses a semantic consistency regularization between the relatedness measured by node states and the evidence given by their true interaction texts. This objective pushes nodes that are structurally close and semantically related to share similar embeddings, which meets our principle for utilizing edge text content for graph embedding.

3.4 Summary of Model Architecture

Putting all components together, we realize the proposed CGAT in a unified graph autoencoder framework; and Figure 4 illustrates how local graph structure and edge content are jointly encoded in one channel-aware attention layer.

We only illustrate the workflow of one layer in our channel-aware attention mechanism; but it generalizes to all other layers. The input is a set of nodes with states $\boldsymbol{h}^{(l)}$ and observed edges among them. Each edge is associated with one or multiple documents, and here we only show one document per edge for simplicity. We first model channels from edge text content. For document $\boldsymbol{w}_{12}$ between nodes 1 and 2, we encode it as a distribution over topics using the topic inference network $q(\theta|\boldsymbol{w}_{12})$ regularized by the prior from node states $\Phi\cdot(h_1^{(l)}+h_2^{(l)})$. We obtain the channel bandwidths λ_{12} according to Eq (3), and the same on other edges, which is highlighted by the first yellow arrow in Figure 4. The inferred bandwidths are then used to obtain the node states for the next layer following Eq (1). As shown in the blue arrow, node 1 updates its state as $h_1^{(l+1)}$ by aggregating its neighbors' information via channel-aware attention. The last layer outputs the node embedding u_1. To jointly learn the embedding of each node and the topic posterior of each edge document, the decoder is optimized to reconstruct the graph structure and the edge textual content following the objective function Eq (9), indicated by the fourth right arrows.

4 EVALUATION

We evaluate the proposed CGAT model on two real-world graph datasets, Yelp and StackOverflow, which provide a wealth of edge text content. A comparative evaluation against a wide variety of graph embedding baselines is performed on the link prediction task to evaluate the models' ability in preserving network structure. We also evaluate the quality of the learned node embeddings in recovering unseen edge text content. Qualitative analysis maps user and topic embeddings to a 2-D space, which demonstrates the improved descriptive power gained from modeling edge textual semantics in CGAT. Finally, a comprehensive case study illustrates hierarchical effect of the semantic channels across different layers for graph embedding learning.

4.1 Experiment Settings

• **Datasets.** We utilized two large publicly available graph datasets associated with rich edge text content for evaluation. 1) ***Yelp***[2] is a bipartite graph between users and items, where 10,192 user nodes

[2]Yelp dataset challenge. https://www.yelp.com/dataset

and 7,694 restaurant item nodes are connected by 641,938 review documents (one review per edge). In average, the node degree is 60.02, and the number of words per review is 24.09. 2) ***StackOverflow***[3] is a social network with 19,016 user nodes, together with 297,238 discussion posts. We utilize the "*reply-to*" interaction in the discussion thread to construct edges between two user nodes, and each interaction attaches one post on the edge. This relation suggests implicit social connections among users based on their expertise and technical interest. In average, the node degree is 15.46, and the number of posts per edge is 1.09 with 22.14 words per post.

On each dataset, we construct its vocabulary containing 2,500 unigram and bigram text features selected by Document Frequency. The node features are constructed by aggregating the bag of words from all documents related to each node. This creates a fair comparison for baselines that do not model edge content, and the performance difference can then be attributed to the design of content modeling on edges. We randomly split the edges linked with each node into 5 folds for cross validation in all the reported experiments.

• **Baselines.** The proposed CGAT is compared against a wide variety of graph embedding models: 1) **GraphSage** [17] uniformly passes information through edges without utilizing node and edge content or features to attend neighboring nodes. 2) **GAT** [30] incorporates node content to reweigh neighboring nodes, and aggregates information from neighbors via a self-attention mechanism. 4) **MNE** [34] utilizes edge type information by constructing node embedding from one base embedding and a set of auxiliary embeddings for each type of relation. 5) **GATNE** [8] also considers edge type and generalizes MNE to the information aggregation framework by self-attention. 3) **EAGCN** [14] directly multiplies general edge feature vectors with the self-attention coefficients to attend neighbors for each feature on edges.

These models vary in how they model edge content. GraphSage and GAT do not consider edge content at all. MNE and GATNE model multi-relational graphs where nodes are connected by multiple types of edges. And they require the edge type to be given. Adapting to our scenario, each edge is assigned with the topic of the largest probability by a pre-trained topic model [5]. Consequently, MNE and GATNE become two-step solutions that model edge text content by a separate model, while our method jointly learns topics and optimizes node embeddings end-to-end. EAGCN incorporates bag-of-word edge feature vectors, but fails to explicitly model the semantic information underlying text (e.g., topics as CGAT does).

To demonstrate the effectiveness of each component in CGAT, we also construct two variants of it by disabling one component at a time: 1) **CGAT-Cnv** removes the convolutional aggregation layers (indicated by the blue arrows in Figure 4), and learns the node embeddings by joint topic and graph structure modeling. 2) **CGAT-Atn** omits the channel-aware attention layer and uniformly passes information from neighbors, which is essentially a rigid combination of GraphSage and a VAE-based topic model.

To evaluate model's predictive power on text content, we also compare with the classic topic model, **LDA** [5], which uses a global Dirichlet prior to model the topic distribution in all documents. We also compared with **AVITM** [27] which uses autoencoding variational Bayes to approximate the posterior distribution of LDA.

[3]StackOverflow. http://stackoverflow.com

• **Details of experiment setup.** All GCN-based models are trained in a minibatch manner following [17] with $L = 2$ layers. For each node in a batch, a fixed number of neighbors and non-neighbors are sampled. We set neighborhood sample size to $S_1 = 25$ and $S_2 = 10$ for layer 1 and 2 respectively, and non-neighborhood size to $\bar{S}_1 = \bar{S}_2 = 20$ for both layers, along with all corresponding edge text documents. The embedding size is set to $m_1 = m_2 = 128$ across all layers for all models. The number of channels is set to $K = 15$ by default. The per-batch complexity of CGAT is $O(\sum_{l=1}^{L} KS_l m_{l-1} m_l + KD(m_{l-1} + |\mathcal{V}| + K))$, where D is the largest number of document per edge and $|\mathcal{V}|$ is the size of vocabulary. The complexity of EAGCN, which also models edge content, is $O(\sum_{l=1}^{L} K' S_l m_l m_{l-1})$, where K' is the dimension of edge feature and $K' = |\mathcal{V}|$ in our case. This shows the efficiency of modeling topic on edge compared with feeding in raw text features. Both source code and our evaluation datasets are made available [4].

4.2 Link Prediction

Link prediction is a widely used task to evaluate the quality of learnt node representations. The link between two nodes is predicted based on the similarity between their embeddings, which formulates the task as a ranking problem to retrieve the most relevant nodes. Each time we held out one fold of edges and the associated text documents as the testing set, and utilized the remaining graph for model training. To construct a candidate set for ranking, we randomly selected negative edges (irrelevant nodes) with an equal number of observed positive edges for each testing node. The quality of ranking is measured by three metrics: mean average precision (MAP), normalized discounted cumulative gain (NDCG) and scaled mean reciprocal rank (MRR) which introduces a scaling factor δ for large candidate pool following [33]. We set the scaling factor to 5 and 1 for Yelp and StackOverflow respectively, since Yelp has a larger candidate pool due to its denser connectivity.

To better analyze the utility of the finer-grained information aggregation introduced in CGAT, we report the prediction performance on nodes with different structural properties. Specifically, we separate the testing nodes into three groups by their normalized degree centrality c, which measures the popularity of nodes. A larger c means the node is more popular and central in the graph; on the contrary, a node with a smaller c is less active. Since information passes through edges in GCN-based models, c also indicates the density of information exchange on different nodes. We categorize nodes as light, medium and heavy by the corresponding thresholds of c. In Yelp, since the nodes are more densely connected, the threshold of c is set to 0.001 and 0.01, while in StackOverflow, it is set as 0.0005 and 0.005 accordingly.

The prediction performance of all models on the two datasets is summarized in Table 1 and Table 2 respectively, where CGAT consistently achieved the best performance on both datasets. Comparing vertically across baselines, we can observe the significance of utilizing edge text content for fine-grain information aggregation. The structure-only method GraphSage performed worse than GATNE and EAGCN, because it cannot utilize edge content to differentiate neighboring nodes. CGAT further outperformed these content-enhanced models, which proves the effectiveness of jointly learning

[4]CGAT. https://github.com/Louise-LuLin/topic-gcn

Table 1: The performance comparison of link prediction on Yelp under different metrics and node groups.

Models	Light User ($c < 0.001$)			Medium User ($0.001 \leq c \leq 0.01$)			Heavy User ($c > 0.01$)		
	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
GraphSage	0.653 ± 0.023	0.472 ± 0.026	0.796 ± 0.016	0.672 ± 0.025	0.487 ± 0.029	0.814 ± 0.014	0.689 ± 0.020	0.497 ± 0.025	0.822 ± 0.015
GAT	0.684 ± 0.025	0.489 ± 0.027	0.812 ± 0.015	0.700 ± 0.024	0.505 ± 0.031	0.833 ± 0.013	0.731 ± 0.025	0.522 ± 0.030	0.841 ± 0.013
MNE	0.577 ± 0.044	0.398 ± 0.046	0.719 ± 0.029	0.622 ± 0.042	0.413 ± 0.045	0.742 ± 0.028	0.634 ± 0.040	0.429 ± 0.042	0.751 ± 0.027
GATNE	0.669 ± 0.030	0.478 ± 0.028	0.805 ± 0.019	0.687 ± 0.027	0.494 ± 0.029	0.825 ± 0.018	0.707 ± 0.025	0.508 ± 0.028	0.834 ± 0.017
EAGCN	0.695 ± 0.024	0.506 ± 0.025	0.827 ± 0.016	0.714 ± 0.025	0.518 ± 0.027	0.845 ± 0.015	0.749 ± 0.023	0.540 ± 0.027	0.858 ± 0.015
CGAT	**0.728±0.022**	**0.527±0.024**	**0.849±0.016**	**0.736±0.024**	**0.543±0.027**	**0.869±0.016**	**0.772±0.024**	**0.564±0.027**	**0.882±0.014**
CGAT-Cnv	0.626 ± 0.032	0.441 ± 0.033	0.763 ± 0.024	0.649 ± 0.030	0.458 ± 0.032	0.788 ± 0.023	0.672 ± 0.029	0.472 ± 0.029	0.796 ± 0.022
CGAT-Atn	0.677 ± 0.024	0.485 ± 0.026	0.811 ± 0.016	0.698 ± 0.025	0.502 ± 0.029	0.834 ± 0.018	0.726 ± 0.026	0.520 ± 0.029	0.838 ± 0.016

Table 2: The performance comparison of link prediction on StackOverflow under different metrics and node groups.

Models	Light User ($c < 0.0005$)			Medium User ($0.0005 \leq c \leq 0.005$)			Heavy User ($c > 0.005$)		
	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
GraphSage	0.721 ± 0.042	0.286 ± 0.041	0.820 ± 0.022	0.733 ± 0.036	0.297 ± 0.031	0.838 ± 0.019	0.744 ± 0.033	0.312 ± 0.029	0.852 ± 0.018
GAT	0.738 ± 0.039	0.298 ± 0.040	0.834 ± 0.021	0.748 ± 0.036	0.307 ± 0.036	0.851 ± 0.019	0.763 ± 0.031	0.327 ± 0.030	0.870 ± 0.019
MNE	0.645 ± 0.056	0.210 ± 0.057	0.745 ± 0.034	0.679 ± 0.051	0.225 ± 0.053	0.772 ± 0.033	0.701 ± 0.048	0.251 ± 0.045	0.788 ± 0.030
GATNE	0.725 ± 0.040	0.291 ± 0.044	0.828 ± 0.026	0.742 ± 0.042	0.303 ± 0.411	0.848 ± 0.253	0.756 ± 0.039	0.322 ± 0.036	0.867 ± 0.023
EAGCN	0.740 ± 0.040	0.302 ± 0.041	0.841 ± 0.021	0.757 ± 0.037	0.318 ± 0.034	0.861 ± 0.020	0.772 ± 0.030	0.337 ± 0.030	0.872 ± 0.020
CGAT	**0.759±0.038**	**0.322±0.039**	**0.859±0.020**	**0.771±0.036**	**0.334±0.035**	**0.878±0.020**	**0.793±0.032**	**0.358±0.027**	**0.897±0.018**
CGAT-Cnv	0.691 ± 0.041	0.242 ± 0.044	0.792 ± 0.027	0.706 ± 0.034	0.274 ± 0.039	0.808 ± 0.232	0.725 ± 0.030	0.284 ± 0.031	0.822 ± 0.023
CGAT-Atn	0.746 ± 0.038	0.305 ± 0.037	0.844 ± 0.022	0.753 ± 0.038	0.314 ± 0.036	0.857 ± 0.022	0.776 ± 0.030	0.341 ± 0.028	0.879 ± 0.017

latent semantics and structural embeddings from content-enriched edges. CGAT-Atn, combining graph structure and edge content without attention mechanism, is distinctly worse than CGAT, which proves the importance of our channel-aware attention layer. The advance of convolution operation is also clearly observed in this experiment: MNE without the convolution layers was largely worse than GATNE, even if they utilized the same information in graph; CGAT-Cnv which disables the convolution operation performed even worse than the baselines without using edge content.

When comparing horizontally across different groups of nodes, we can observe a larger performance gap on heavy nodes between CGAT and the baselines that do not use edge content, i.e., GraphSage and GAT. This seems to be counter-intuitive, since incorporating edge content is expected to help more on sparse nodes with less structural information. On the contrary, better performance on heavy nodes exactly aligns with our proposed principle of utilizing edge content to differentiate the influence from neighbors. Heavy nodes tend to connect with a variety of neighbors with distinct topical interests; therefore, aggregating information uniformly without differentiating their semantic relatedness would result in inaccurate embeddings. The elaborated design of the channel-aware attention layer in CGAT addresses this issue by attending on different semantics inferred from edge content, such that the embeddings can be structurally and semantically separated.

4.3 Content Prediction

Recall that the node embeddings learned by CGAT can be projected to the topic space by $\Phi \cdot (u_i + u_j)$, which can serve for content prediction on the predicted links, e.g., what content might be generated once this link is established. We compared CGAT with topic models

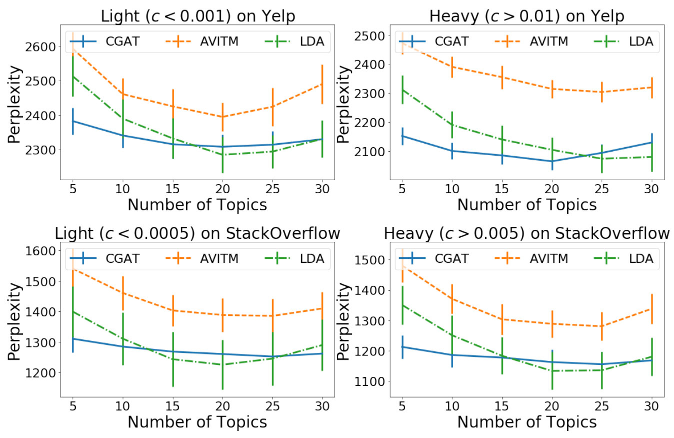


Figure 5: Perplexity comparison on Yelp and StackOverflow.

by their *perplexity* on the held-out edge text documents. Formally, the perplexity for a set of documents is calculated as follows [5]:

$$perplexity(\mathcal{D}_{test}) = \exp\left(-\frac{\sum_{d \in \mathcal{D}_{test}} \log p(\mathbf{w}_d)}{\sum_{d \in \mathcal{D}_{test}} |\mathbf{w}_d|}\right)$$

where $p(\mathbf{w}_d)$ is the likelihood of document d given by the model. A lower perplexity indicates a better content generation quality.

Figure 5 reports the mean and variance of perplexity from 5-fold cross validation over different topic size K. On each dataset, we again summarize the comparisons with respect to light and heavy nodes. CGAT achieved comparable performance with the original LDA model with Dirichlet prior. And it consistently outperformed AVITM that uses a similar variational autoencoder for posterior

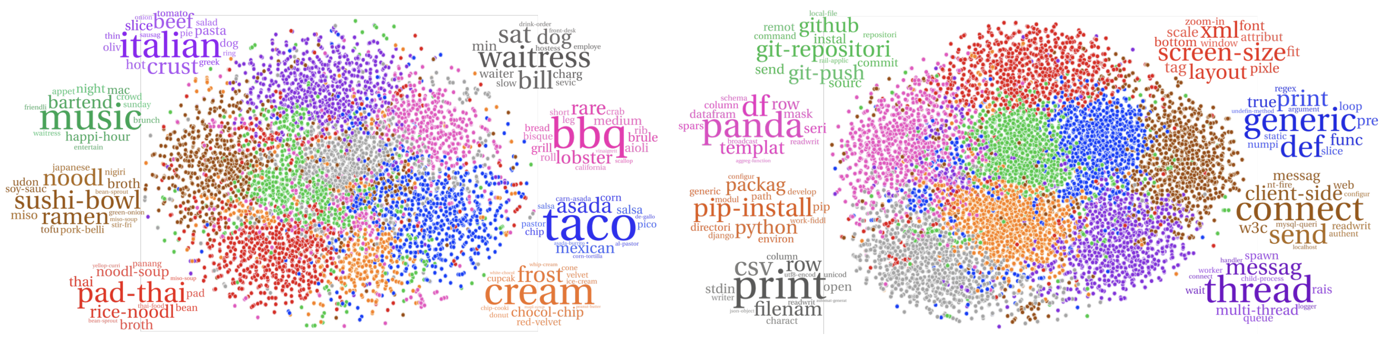


Figure 6: Visualization of the learnt node embeddings on Yelp (left) and StackOverflow (right). Node color denotes the topic that this node (i.e., user) is closest to in the projected topic space. The word cloud shows the most representative words of the topic with the same color.

inference but with a global prior for topic distribution across all documents. CGAT uses node embeddings pertaining to each edge to infer its prior topic distribution, which provides local context on a per-edge basis. This leads to its improved content prediction on the edges. This also proves the learned node embeddings preserve semantic relatedness among nodes, which leads to its potential for an interpretable link prediction. Moreover, CGAT demonstrates stable performance with respect to different topic sizes, which verifies the benefit of jointly modeling topics with graph structure.

4.4 Visualization of Node Embeddings

To analyze the quality, especially the inferred semantics, of the learned node embeddings by CGAT, we visualize its embeddings in a 2-D space using the t-SNE algorithm in Figure 6. For illustration purpose, we assign each node to its closest topic, i.e., $\arg\max_k(\phi_k \cdot u)$; and we mark nodes sharing the same interested topic with the same color. We also plot the most representative words under each topic in $p(w|\beta_z)$ learned from CGAT with the same color of the corresponding set of nodes.

We can observe on both datasets that nodes sharing similar topical interests are closely clustered, which verifies the realization of the proposed principle: semantically similar nodes should be closer in the embedding space. Meanwhile, we can easily interpret the semantic meaning of each embedded node using the jointly learned topics. On Yelp, it is rather direct to distinguish different user groups' preferred cuisine types: e.g., Italian (in purple) v.s., Mexican (in blue); similarly on StackOverflow, we can recognize users' expertise in programming: e.g., network communication (in brown), v.s., interface design (in red). Another interesting finding is that the distance between nodes characterizes the relatedness between topics. For example on StackOverflow, users interested in network communication (in brown) are closer to those who focus on multithreading (in purple), than those on dataframes (in pink); and on Yelp, users preferring Japanese food (in brown) get closer to those who prefer Thai food (in red) than those in favor of Mexican food (in blue). This again demonstrates that jointly learning topics with node embeddings on top of a graph structure not only introduces semantic meaning into the learnt embeddings, but also preserves structural information of the graph, such that topics manifested by connected nodes tend to be more related.

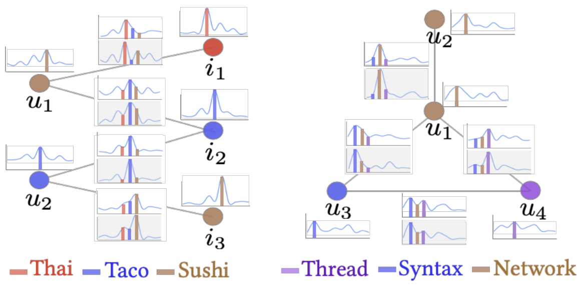
4.5 Case Study about Inferred Attention

Finally, we use a case study in Figure 7 to visualize the computations in channel-aware attention layer for a deeper understanding of it. We zoom into several nodes and edges chosen from the Yelp and StackOverflow graphs respectively. Each node is tied with the topic distribution given by $\Phi \cdot u$, and is colored by the topic of the largest probability. Recall that we stacked $L = 2$ layers and both layers are used to reconstruct the edge text content; and thus we visualize both sets of inferred channel bandwidths on each edge: the upper illustration $\Lambda^{(2)}$ is for the output layer, and the lower one $\Lambda^{(1)}$ is for the first aggregation layer. Meanwhile, we select two edges from each graph and excerpt the ground truth text content. The color denotes the topic from which the word is generated.

The semantic relatedness between edge and connected nodes is quite consistent. If we look at each pair of connected nodes, we can find dominant channels on the edges, which pass more information for corresponding topics, and take major parts in forming embeddings for the end nodes. For example on Yelp, user u_1 and item i_1 are connected and channel *Thai* has the largest bandwidth for information passing, and thus when aggregating neighbors' states for user u_1, item i_i mostly influences u_1 on topic $Thai$; similarly, u_2 and i_2 exchange more information through the major channel *Taco*. The text content on edges (u_1, i_1) and (u_2, i_2) illustrates the dominant topics, which verifies the learned channel. A more interesting observation is that though the channel bandwidths on different layers show a similar distribution, they represent different levels of semantic relatedness between nodes. For each edge, the bandwidths of the first layer (i.e., $\Lambda^{(1)}$) presents a steeper shape with more emphasis on those dominant topical channels; while the upper one for the output layer $\Lambda^{(2)}$ demonstrates a flatter shape over all channels in general. This indicates that the convolution operation aggregating neighbors' information may produce a community-level embedding for the second layer, which can result in more general channel bandwidths with less individuality.

5 CONCLUSION AND FUTURE WORK

In this paper, we enhanced graph representation learning by utilizing the latent semantic information carried in edge content. We proposed a unified graph autoencoder framework named CGAT to profile each edge as a mixture of semantic-driven channels inferred



Edge	Excerpted Text on Edge
(u_1, i_1)	We tried rice noodle. The pho is more of a Thai version. Pad se ew was among the best BF has had.
(u_2, i_2)	The al pastor taco is where it's at my friends. The tortillas are handmade. Salsa bar is bomb, so many choices!
(u_1, u_3)	You forgot to enclose the command with single quotes. The shortcut syntax should produce a message with 2 extra headers.
(u_1, u_4)	Make sure background worker threads have access to a Context object (refer to the service-activity communication section.

Figure 7: (Left) A case study of how information is passed through topical channels on Yelp and StackOverflow. Each node is attached with a topic distribution $\Phi \cdot u$. Each edge is tied with two sets of channel bandwidths: $\Lambda^{(1)}$ for the first layer (shaded) is beneath $\Lambda^{(2)}$ for the output layer. (Right) Two example edges for each graph with excerpted ground-truth text content.

from edge text content, and guide information aggregation via a channel-aware attention mechanism. The learned node embeddings are demonstrated to be predictive and interpretable.

As our future exploration, we are especially interested in the community effect of convolution operation shown in our case study. At the upper level convolution layers, the information is aggregated from farther neighbors which results in node representations encoding both individual- and community-level information. It is necessary to explore such a hierarchical structure for multi-level interpretability of learnt representations. Moreover, currently we take a static view of the graph structure and edge content, i.e., the graph is considered as static and fixed. Given the graph is forever evolving, so is its edge content, it is important for us to extend CGAT to model the dynamics on graph, both structure and edge content. This gives us a new opportunity to understand graph data more deeply.

ACKNOWLEDGMENTS

This paper is based upon work supported by the National Science Foundation under grant IIS-1553568 and IIS-1718216.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. 2018. Watch your step: Learning node embeddings via graph attention. In *NeurIPS*. 9180–9190.
[2] Charu C Aggarwal, Haixun Wang, et al. 2010. *Managing and mining graph data*. Vol. 40. Springer.
[3] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[4] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.
[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[7] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *TKDE* 30, 9 (2018), 1616–1637.
[8] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *SIGKDD*. 1358–1368.
[9] Pengfei Chen, Weiwen Liu, Chang-Yu Hsieh, Guangyong Chen, and Shengyu Zhang. 2019. Utilizing edge features in graph neural networks via variational information maximization. *arXiv preprint arXiv:1906.05488* (2019).
[10] Robert B Cialdini and Noah J Goldstein. 2004. Social influence: Compliance and conformity. *Annu. Rev. Psychol.* 55 (2004), 591–621.
[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*. 3844–3852.
[12] Frederik Diehl. 2019. Edge Contraction Pooling for Graph Neural Networks. *arXiv preprint arXiv:1905.10990* (2019).
[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*. JMLR. org, 1263–1272.
[14] Liyu Gong and Qiang Cheng. 2019. Exploiting Edge Features for Graph Neural Networks. In *CVPR*. 9211–9219.
[15] Lin Gong, Lu Lin, Weihao Song, and Hongning Wang. 2020. JNET: Learning User Representations via Joint Network Embedding and Topic Embedding. In *WSDM*. 205–213.
[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.
[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[18] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[19] Diederik P Kingma, Max Welling, et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392.
[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[21] Lu Lin, Lin Gong, and Hongning Wang. 2019. Learning Personalized Topical Compositions with Item Response Theory. In *WSDM*. 609–617.
[22] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.
[23] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *WSDM*. 601–610.
[24] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.
[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.
[26] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, 593–607.
[27] Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488* (2017).
[28] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. 2004. Link prediction in relational data. In *NeurIPS*. 659–666.
[29] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *ACL*. 1722–1731.
[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[31] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.
[32] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. 2020. Relation Learning on Social Networks with Multi-Modal Graph Edge Variational Autoencoders. In *WSDM*. 699–707.
[33] 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 *SIGKDD*. 974–983.
[34] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding. In *IJCAI*, Vol. 18. 3082–3088.