Collective Text Classification Using Topic-enhanced Latent Textual Links

Yanbang Wang
ywangdr@cs.stanford.edu
Stanford University

Carl Yang
j.carlyang@emory.edu
Emory University

Weixin Liang
wx@stanford.edu
Stanford University

Pan Li
panli@purdue.edu
Purdue University

Jiawei Han
hanj@illinois.edu
Univ. of Illinois at Urbana-Champaign

ABSTRACT

Collective text classification studies the problem of classifying a collection of documents together by exploiting their inter-document relationships such as paper citations or web hyperlinks. Previous studies typically assume that such relational data are informative for prediction, and use them in ways that highly trust their usefulness. Such assumption and reliance, however, make them easily suffer when the given relational data are actually less informative. We propose that an effective solution is to enrich more types of relational information and to consider various relational information simultaneously when modeling the text. Our work explores this possibility by exploiting latent but rich *textual links* among documents that can be explicitly formulated by just analyzing the text. In our framework, we first construct a heterogeneous text network to instantiate and encode various semantic relationships among documents bridged through topics and n-grams. Then, we design a Graph Edge-Attention Network to embed the structural heterogeneity and model the complex interactions between various text entities and their relationships. Experiments on multiple benchmark datasets show that our effective usage of latent textual links helps significantly improve classification performance over the state-of-the-art baselines. Various ablation studies and hyperparameter analysis provide further insight into interesting properties of the latent textual links we construct.

CCS CONCEPTS

• **Information systems** → **Document representation**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

text classification, collective classification, textual links, GNN

ACM Reference Format:
Yanbang Wang, Carl Yang, Weixin Liang, Pan Li, and Jiawei Han. 2021. Collective Text Classification Using Topic-enhanced Latent Textual Links. In *Proceedings of WWW '21: International World Wide Web Conference (WWW '21).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Text classification, which aims to assign categorical labels to a collection of documents, is a fundamental problem that prototypes many NLP tasks. Recent years have witnessed great advancement in this area by applying deep models to generate high-quality representation for each individual piece of text [6, 14, 39].

Orthogonal to that direction, collective text classification (CTC) directs another line of research which also proves tremendous success [33, 34, 36, 41, 43]. Instead of focusing on how to extract better features for each individual document, CTC methods classify multiple documents collectively by exploiting the relational information among them, usually through the abstraction of text-rich information networks. For example, academic papers are classified by using their citation information [8], web pages by referring to their hyperlinks [30], tweets by utilizing people's friend relationships [7]. Central to CTC's success is that the given linkage information helps guide the collection and aggregation of more text samples relevant to the target text to predict, so that the input feature space for each target text is enlarged to group-level, which may reduce both bias and variance of the prediction.

However, the advantage of CTC methods also entails a critical intrinsic weakness, which is that their performance gain heavily relies on the quality of the pregiven inter-document links [12]. Most existing works assume that those inter-document links are readily available and use them in ways that significantly trust their fidelity and usefulness for the task. For example, methods that model the joint probability distribution of edges like TADW [41], CANE [36], and paper2vec [8] are all trained on the given links. Graph Neural Network (GNN) based methods such as GCN [17], GAT [37], and their applications to CTC [13, 19] also aggregate information from a localized neighborhood exclusively defined by the given links. A consequence of such reliance is that the performance gain can be seriously undermined when those pregiven links are less relevant to the task or contain too much noise.

We propose that one effective solution to the aforementioned problem is to extensively enrich the link types that we consider. In other words, we can further relate documents by analyzing just their text data and by directly extracting their latent but rich semantic relationships from their text. We call such latent semantic relationships *textual links*. Textual links are very different from the aforementioned links as citation, hyperlinks, or friendship, which by their nature describe certain semantic-free relationships, and we call those links *non-textual links*. While the usage of textual links is largely under-explored previously, we believe that it also plays a critical role in defining a properly connected network of text entities that can significantly benefit classification tasks.

Problems with existing works. There were sporadic attempts that implicitly entail the spirit of leveraging textual links, such as Text GCN [43] and PTE [33]. However, none of them well formulate and address the critical question as to how to best *define* and *leverage* textual links among documents, which is the real key to a successful CTC method.

Both PTE and Text GCN share a similar network construction scheme as the following: they connect a document (node) to *all* its word nodes, and connects two word nodes by co-occurrence. Such construction, however, has several drawbacks: (1) The textual relationships between documents bridged only by shared words is usually very weak and noisy, and so using such links as the backbone for the network is insufficient for making accurate classification. (2) In order for the word co-occurrence links to be valid, both methods require using (almost) full vocabulary as word nodes, leading again to strong noise and high computation complexity. (3) After the network construction step, both methods then proceed to the representation learning step which learns document node representations for prediction. The two steps, however, are conducted in a completely detached manner– their construct a network in a fully unsupervised way, and their supervised representation learning step has to accept whatever is hardcoded into their constructed network as input. The lack of integration

In short, both works share several inherent drawbacks including a sub-optimal construction scheme of the information network and a computation framework that is less effective in modeling the interaction of heterogeneous textual relationships.

Our Method. Central to what we propose are three things integrated into a unified framework that makes the best of textual links for classification. We first construct a well-connected heterogeneous text network to encode the rich latent inter-document relationships in a compact and unified manner. The network (Fig. 1) involves three types of nodes: document, topic, and n-gram, and four types of relationships among them. To determine the topic-associated structural weights which are critical in shaping the connectivity of our constructed network, we further discuss two strategies of generating them with different advantages: one strategy uses a pre-trained topic model to initialize them, and the other automatically learns them jointly with our representation learning framework by using variational components. Finally, we present a Graph Edge-Attention Network (GEAT) to learn representations over our constructed network. GEAT takes a neat, unified form of message propagation among various text entities to refine their features. However, it still well captures the information heterogeneity with its edge-based attention, which lets the propagation be modulated differently by different relationships.

2 RELATED WORK

2.1 Link-Free Text classification

Most existing works on text classification focus on representing documents without considering any inter-document relationships. They start from using basic word-level features (*e.g.* bag-of-words, n-gram[4], TF-IDF[29]), to using word embeddings for better capturing word meanings such as [22, 25, 27]. Recent years also see the prosperity of deep language models [6, 14, 15, 20, 21, 39] that capture fine-grained dependencies within the document. All this literature relates to our work in that we can use any of them as an upstream feature extractor to generate individual document representations, and then pipe those representations to our framework as initial document node features.

2.2 Collective Text Classification

To exploit the inter-document relationship, collective text classification methods combine the document linkage information with representations of individual documents. They typically construct a homogeneous information network where nodes represent documents and edges represent linkage among the documents, *e.g.* citations, hyperlinks. To generate representations for the document nodes, two classes of network embedding methods are mostly used: (1) Methods that model the joint probability of links such as Deepwalk [28], LINE [34], and their adaptation to further incorporate text attributes such as TADW [41], paper2vec [8], CANE [36], and PTE [33]; (2) Graph Neural Networks such as GCN [17] and GAT [37], and Text GCN [43] which adapts GCN for text embedding.

All methods above require semantic-free document links as input, except Text GCN and PTE, which construct extra document links from the text data. Therefore, the two methods are closest to what we propose. They share a similar network schema that connects a document (node) to *all* its word nodes, and connects two word nodes by co-occurrence. Such construction, however, has several drawbacks: (1) The textual relationships between documents bridged only by shared words is usually very weak and noisy, and so using such links as the backbone for the network is insufficient for making accurate classification. (2) In order for the word co-occurrence links to be valid, both methods require using (almost) full vocabulary as word nodes, leading again to strong noise and high computation complexity. (3) Both methods' representation learning step does little to capture the heterogeneity of different relationships and node entities, and are completely detached from their network construction. Our ablation studies in Sec.3 provide further empirical evidence.

2.3 Topic Modeling

Our work relates to topic modeling by its core concept of "topic", which we use to enhance the structuralization process in our classification framework. Classic topic models include LSA[5], pLSA [10], LDA[2], HDP[35], *etc.* Recent years also see a plethora of works on neural topic modeling using variational and adversarial techniques such as [23, 24, 38]. Topic models have been extensively used in various applications that require the modeling of community structure of a collection of documents [9, 26, 31]. The topic vectors can also serve as document representations with decent prediction power.

3 PRELIMINARIES

In this section, we first formalize the distinction between *textual links* and *non-textual links* as foundation of our discussion. Next, we define the collective text classification problem that we consider. Finally, we introduce several basic concepts in topic modeling, which is widely used as a principled and elegant way to obtain structural insight into a text corpus.

3.1 Textual and Non-textual Links

We define **textual links** as links which relate two documents and are derived by analyzing just the documents' text. They can be either direct or indirect, meaning that textual links can exist either between two documents, or can chain two documents through a series of intermediate text entities such as words or topics. For example, document A can indirectly relate to document B because they share a certain keyword. In this case the textual link instantiates the <contain> relationship between a document and a word. This definition endows textual links with strong semantic expressiveness. As opposed to textual links, **non-textual links** are links which relate two documents and can *not* be derived by just analyzing their text. Classic examples are paper citations and webpage hyperlinks, which have been studied much more extensively.

3.2 Problem Definition

The problem of CTC typically provides two input elements:

(1) A collection of documents $D = \{d_1, \dots d_{N_d}\}$, where $N_d = |D|$ is the number of documents. Each document contains a sequence of words. A subset $D_{train} \subseteq D$ is labeled, in which each $d_i \in D_{train}$ is assigned one of the C class labels from $\{1, \dots, C\}$. $D_{test} = D \backslash D_{train}$.

(2) A set of non-textual links between documents. The links can be represented as an adjacency matrix $A \in \mathbb{R}^{|D|\times|D|}$, where $A_{dd'}$ denotes strength of the relationship between documents (indexed) d and d', and $A_{dd'} = 0$ if they are not related. Note that though we primarily consider the case of undirected links for simplicity, generalization to directed links is straightforward: one can simply use positive and negative signs to distinguish the two directions.

Our goal is to assign a class label to each document $d \in D_{test}$.

3.3 Latent Dirichlet Allocation (LDA)

The Latent Dirichlet Allocation is a highly successful topic model that provide scalable, robust, and inductive decomposition of the rich underlying semantic structures of a given corpus. It uses a probabilistic model to depict the generation of documents by the following process:

$$\theta_d^{(lda)} \sim \text{Dirichlet}(\alpha^{(lda)}), \qquad \text{for } d \in D \tag{1}$$

$$z_n^{(lda)} \sim \text{Multinomial}(\theta_d^{(lda)}), \qquad \text{for } n \in [1, N_d] \tag{2}$$

$$w_n^{(lda)} \sim \text{Multinomial}(\beta_{z_n}^{(lda)}), \qquad \text{for } n \in [1, N_d] \tag{3}$$

$\alpha^{(lda)}$ is a parameter vector that shapes the Dirichlet prior for topic allocation, θ_d is the topic allocation vector drawn from the prior, N_d is an auxiliary variable determining the number of terms in d, $z_n^{(lda)}$ is an one-hot indicator variable conditioned on θ_d and is repeatedly drawn for different term w_n, $\beta^{(lda)}$ parametrize the topic-term distribution and $\beta_{z_n}^{(lda)}$ denotes the how topic z_n distributes over terms. We use the superscript $^{(lda)}$ to distinguish the variables with variables in our proposed method.

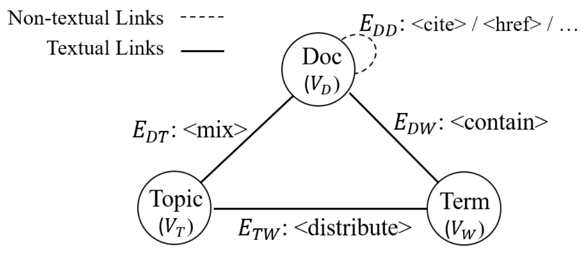

Figure 1: Schema of the constructed heterogeneous text network.

LDA provides a clean and principled prototype of the semantic relationships among various entities (*i.e.* documents, topics, terms) in a corpus. Those relationships are well-encoded by the latent variables and distribution parameters like $\beta^{(lda)}$ and $\theta^{(lda)}$.

4 PROPOSED METHOD

Given the input defined in Sec. 3.2, we first discuss in Sec. 4.1 how we construct a well-connected heterogeneous text network to encode the rich latent inter-document relationships in a compact and unified manner. Then, Sec. 4.2 dives deeper into the computation of some crucial link weights of the constructed network. Finally, Sec. 4.3 presents a Graph Edge-Attention Network which captures the complex interactions among various entities and learns document representations.

4.1 Heterogeneous Text Network Construction

We construct a heterogeneous text network G whose architecture is shown by the network schema in Fig. 1. Our network consists of three types of nodes and four types of links:

Document nodes V_D represent the set of documents in the dataset. Their input node feature matrix $z_D \in \mathbb{R}^{|D|\times M_D}$ are directly piped from the document-level representations extracted by an upstream text feature extractor. In principle, any text feature extractor can be used — from the classic document features such as bag-of-words or TF-IDF, to the more advanced neural architecture like BERT, both of which are evaluated in our experiment.

Topic nodes V_T instantiate the set of topics T which we want to explicitly extract and formulate from the corpus. The number of topic nodes $|V_T|$ is equal to the topic number $|T|$ which is a hyperparameter. Their feature matrix $z_T \in \mathbb{R}^{|T|\times M_T}$ is initialized from a trainable embedding lookup table.

Term nodes V_W instantiate a set of selected n-grams from the document collection. While existing literature provides different ways of for n-gram selection [3, 11], in practice we found that filtering criteria based on both TF-IDF and topic scores works well. We will elaborate more on this in Sec. 5. However, we note that the key here is to avoid the usage of (nearly) the entire set of n-grams — this is especially important in a noisy corpus, and in practice we surprisingly found that optimal performance is usually reached with less than 5% of all n-grams. Another advantage of pre-filtering n-grams is that it can significantly scale down our heterogeneous network along with the computation complexity for representation learning. Similarly for term nodes, their feature matrix $z_W \in \mathbb{R}^{|W| \times M_W}$ is also initialized from a trainable embedding lookup table.

Document-Document Links E_{DD} encode the non-textual relationship and are directly mapped from the input non-textual links. Formally speaking, the induced subgraph of G from all its document nodes D, G_{DD}, has its connectivity fully expressed by the adjacency matrix A defined in Sec. 3.2. To encode the link, we use one-hot vector to denote the relationship's type, and further weigh it by the relationship's strength, *i.e.* $z_{dd'} = [A_{dd'}, 0, 0, 0]^\intercal, \forall\ (d, d') \in E_{DD}$.

Document-Term Links E_{DW} encode the *<contain>* relationship between a document and an n-gram term in that document. To encode the link, we use one-hot vector to denote the relationship's type, and further weigh it by the n-gram's corresponding TF-IDF value w.r.t the document: $z_{dw} = [0, A_{dw}, 0, 0]^\intercal,\ \forall\ (d, w) \in E_{DW}$.

Document-Topic links E_{DT} **and Topic-Term links** E_{TW} instantiate the two important structural relationships revealed by the topic modeling: **(1)** a link $(d, t) \in E_{DT}$ exists between a document $d \in D$ and a topic $t \in T$, and represents the *<mix>*ture of topic t in document d; **(2)** a link $(t, w) \in E_{TW}$ exists between a topic $t \in T$ and a n-gram $w \in W$, and represents the how topic t *<distribute>*s over the n-gram w. Both types of links are also associated with weights that parametrize the corresponding topic mixture θ (and θ_{dt} for specific d and t) or word distribution β (and β_{tw} for specific t and w). θ and β are crucial structural weights that shape the connectivity of our network, and we dedicate Sec. 4.2 to their generation. For the time being we assume that their values are ready to use, and proceed to use one-hot encoding for both types of links: $z_{dt} = [0, 0, \theta_{dt}, 0]^T, \forall (d, t) \in E_{DT}$, and $z_{tw} = [0, 0, 0, \theta_{tw}]^\intercal, \forall\ (t, w) \in E_{TW}$.

4.2 Generating Topic-associated Weights

A key unsolved problem we are left with from Sec. 4.1 is the derivation of the topic-associated link weights θ and β. As mentioned, they are crucial structural weights that shape the connectivity of our constructed network, and thus will significantly affect the learned representations later. Therefore, extra care should be given to ensure that they are computed to reflect their intended semantic meanings with best fidelity and predictive power. Here we discuss two well-principled strategies.

4.2.1 *Initialized with a Pretrained Topic Model*.

One straightforward method of determining θ and β is to directly map them from a topic model. We take LDA as an example to illustrate this process:

$$D, W, |T| \xrightarrow{\text{LDA}} \theta^{(lda)}, \beta^{(lda)} \xrightarrow{\text{Init.}} \theta, \beta \tag{4}$$

Given a collection of documents D, the n-grams W, and the topic number $|T|$, we include a preprocesssing step which fits a LDA model to D to learn the topic-word distribution parameter $\beta^{(lda)}$, and to infer the document-topic mixture latent variable $\theta^{(lda)}$. We then use the derived $\theta^{(lda)}$ and $\beta^{(lda)}$ to respectively initialize the θ and β used in link encoding. During the training, we do not freeze θ and β, allowing them to be fine-tuned with gradient descent together with other parameters. This gives our **LDA-pt** strategy, which suggests that we use a **pre**t**r**ained LDA model to initialize some link weights.

The **LDA-pt** strategy has the advantage that the weights can be initialized in a well-defined manner by the topic model while also enjoying some level of flexibility to be further refined later by the supervised signals. However, it may not always be the case that the pretrained topic model can yield predictive topic distributions. Therefore, we propose a second variant, the **VTG** strategy, which allows θ and β to be jointly trained with our graph neural architecture proposed later in 4.3 to explicitly generate predictive topic distributions for the classification task.

4.2.2 *Joint Training with Variational Topic Generation*.

The basic idea of the **VTG** strategy is that it employs several neural variational components to simulate the generative and inference process of topic modeling. This process allows θ and β to be learned by error back-propagation and gradient descent, which can be carried out jointly with representation learning over the whole network proposed in 4.3. It consists of two components for document generation and variational inference, respectively.

Generation. For each document $d \in D$, our generative process takes in a random vector sampled from isotropic Gaussian prior, *i.e.* $\phi_d \sim N(\mu_d, \rho_d^2)$, and generate the set of selected n-grams W. The process starts by simulating the Dirichlet prior to derive the simplex latent variable $\theta_d \in \mathbb{R}^{|T|}$ (here we use θ_d to denote the mixture of document d over topics T). We use the Gaussian-softmax function for the simulation which is similar to[32]:

$$\theta_d = g(\phi_d) = \text{softmax}(Q_1 \phi_d + b_1) \tag{5}$$

where projection matrix Q_1 and bias b_1 are parameters to be learned. Joint probability of the selected n-grams is then given by:

$$p(W|\phi_d) = p(W|\theta_d) = \prod_{w \in W} (w|\phi_d) = \prod_{w \in W} \sum_{t \in T} p(t|\theta_d) p(w|t) \tag{6}$$

where $p(t|\theta_d)$ is the probability of selecting topic t given distribution vectorθ_d, which is exactly parametrized by θ_{dt}; $p(w|t)$ is also parametrized by β_{tw}. Together we have:

$$p(w|\phi_d) = \prod_{w \in W} \sum_{t \in T} (g(\phi_d))_t \beta_{tw} \tag{7}$$

Inference. For a given document d, the inference process approximates the true posterior distribution of ϕ_d conditioned on d and the set of topics T. With the re-parametrization trick [16] $\phi_d = \mu_d + \epsilon * \rho_d, \epsilon \sim N(0, 1)$, we infer the posterior using bilinear product attention based on encoding $z_d \in \mathbb{R}^{M_D}$ and $z_T \in \mathbb{R}^{|T| \times M_T}$:

$$p(\phi_d|d, T) \approx q(\phi_d|d, T) = q(\mu_d, \rho_d|d, T) \tag{8}$$

$$\mu_d(d, T) = z_T Q_2 z_d^\intercal; \quad \rho_d(d, T) = \exp(z_T Q_3 z_d^\intercal) \tag{9}$$

where $Q_2, Q_3 \in \mathbb{R}^{M_T \times M_D}$ are learnable bilinear projection matrices.

Learning. We optimize the variational evidence lower bound $\mathcal{L}_d^{(vtg)}$ for document d w.r.t parameters $Q_1, Q_2, Q_3, b, \beta, z_T$:

$$\mathcal{L}_d^{(vtg)} = \mathbb{E}_{q(\phi_d|d,T)}\left[\sum_{w\in W} I_{dw}\log\sum_{t\in T} p(t|\theta_d)p(w|t)\right] - D_{KL}(q(\phi_d|d,T)||p(\phi_d)) \quad (10)$$

where D_{KL} denotes the KL-divergence between two distributions. Here because of the usage of Gaussian prior, D_{KL} can be integrated with a closed form derivation following [16]. I_{dw} is an indicator variable, *i.e.* $I_{dw} = 1$ *iff.* w is a word in document d.

The advantage of **VTG** is that the whole framework can be elegantly trained end-to-end, allowing it to generate topic-associated link weights that desirably have stronger predictive power. Meanwhile, notice that the topic encoding z_T, is also involved as trainable parameters. This bridge further allows the topic generation process and the graph neural network to interact with and benefit from each other, and to refine the topic encoding itself. Finally, from the perspective of regularization, we also point out the KL-divergence term D_{KL} and the Gaussian noise ϵ essentially helps regularize learning of the topic-associated link weights in a principled manner. This can be especially helpful to deal with the strong noise and randomness typically contained in text data.

4.3 Representation Learning in the Network

Finally, we propose a **G**raph **E**dge **At**tention Network (GEAT) to learn representations in the heterogeneous network we constructed. Input to GEAT are the heterogeneous node and edge features : Z_D, Z_T, Z_W, Z_{DD}, Z_{DT}, Z_{TW}, Z_{DW}. We further use lowercase letter z to represent an individual node or link, *e.g.* z_d denotes input node features of the a specific document d, and similarly for other node and link types.

The first thing GEAT does is to align the distribution space of all the node and edge features via linear transformations. That is, for all $d, d' \in D, t \in T, w \in W$ where appropriate:

$$x_d = f_D(x_d), x_t = f_T(z_t), x_w = f_W(x_w) \quad (11)$$

$$e_{dd'} = f_{DD}(x_{dd'}), e_{dw} = f_{DW}(z_{dw}),$$
$$e_{dt} = f_{DT}(x_{dt}), e_{tw} = f_{TW}(x_{tw}) \quad (12)$$

where: $f_\Omega : \mathbb{R}^{F_\Omega} \to \mathbb{R}^{M_V}$, for $\Omega \in \{D, W, T\}$; and $f_{\Omega'} : \mathbb{R}^4 \to \mathbb{R}^{M_E}$,for $\Omega \in \{DD', DW, DT, TW\}$, where M_V and M_E are dimensions of the unified node feature space and link feature space, respectively. The seven f's with different subscripts denote the different linear projection functions.

Eqn.(11) and (12) essentially reduce the heterogeneous network into a homogeneous one while preserving the heterogeneity by different f's. Therefore, from now on we can safely ignore the subscripts of features x and e, and instead use x_i, x_j, and e_{ij}, where the $i, j \in D \cup T \cup W$ are type-free indices.

GEAT then proceeds to carry out message passing among nodes with edge-level modulation. Eqn. (13) and (14) give one layer of GEAT:

$$x_i = x_i + \sigma(H_n \sum_{j\in\mathcal{N}(i)} a_{ij}x_{ij}) \quad (13)$$

$$a_{ij} = \frac{\exp(H_e e_{ij} + b_e)}{\sum_{j'\in\mathcal{N}(i)}\exp(H_e e_{ij'} + b_e)} \quad (14)$$

, where i is the center node, $\mathcal{N}(i)$ is node i's neighbors, $H_e \in \mathbb{R}^{M_E\times h}$, $H_n \in \mathbb{R}^{M_N\times h}$ are two parameter matrices, σ is a non-linear function such as ReLU.

After L layers of GEAT, we obtain representations for all the document, topic and term nodes. Representations for the document nodes are further retrieved and piped to a single-layer perceptron to do classification and learn from the error via the standard Softmax and Cross Entropy loss:

$$\mathcal{L}^{(clf)} = \sum_{d\in D} \text{cross-entropy}(c_d, f_{out}(o_d)) \quad (15)$$

where c_d is the ground truth label of document d, o_d is the output representation from GEAT, f_{out} is a single-layer neural network with softmax. Finally, if the **LDA-pt** strategy in Sec. 4.2 is adopted, we can directly optimize $\mathcal{L}^{(clf)}$ *w.r.t* GEAT's parameters. If **VTG** strategy is adopted otherwise, our optimization objective is the new loss $\mathcal{L}$ given by the weighted sum of the classification loss and the variational loss by Eqn.(16) with $\lambda \in [0, \infty]$:

$$\mathcal{L} = \mathcal{L}^{(clf)} + \lambda \sum_{d\in D} \mathcal{L}_d^{(vtg)} \quad (16)$$

How is relationship heterogeneity captured by GEAT (and why not pte, text gcn)?

5 EXPERIMENT

5.1 Experimental Setups

We implemented two variants of our model: **GEAT-LDA-pt** and **GEAT-VTG**, corresponding to the two weight generation strategies discussed in Sec.4.2. For the rest of this section, we report more details on our baselines, datasets, and training configurations.

5.1.1 *Baselines.*

Our method is compared with 11 baselines that cover a wide range of existing literature. Their detailed tuning process is in Appendix.

• **TF-IDF:** the classic document feature of Term Frequency–Inverse Document Frequency. We pipe it to a 2-layer fully connected neural network for classification. We use that as the most basic baseline.

For all the methods below that do not specify its input document node features (e.g. our method, all GNN-based baselines, TADW, etc.) we also consistently use the basic TF-IDF features as input document node feature. We do so because our study's focus is not on representing document individually, though we also include a special baseline, BERT [21], to showcase the state-of-the-art gain that can be made in that direction for reader's further reference.

• **LDA [2]:** Latent Dirichlet Allocation. We use the topic vector generated by the topic model for each document as its representation. Then we use a fully connected neural network for classification.

• **DeepWalk [28]:** the classic method that learns network embedding by sampling and modeling the pregiven links' joint probability.

• **TADW [41]:** Text-associated DeepWalk, which extends DeepWalk to incorporate text features into the network embedding.

• **paper2vec [8]:** a method that builds a K-nearest-neighbor graph among documents. It then merges with the citation network to learn node embeddings by modeling all the links' joint probability.

• **CANE [36]:** Context-Aware Network Embedding, which learns multiple embeddings for a node when it interacts with different neighbors. Its optimization objective is still to model the pregiven links' joint probability.

• **PTE [33]:** a method that seeks to learn predictive word and document embeddings by using labels to build multiple bipartite heterogeneous networks. It then adapts [34] to learn embeddings.

• **GCN [17]:** Graph Convolutional Network, a representative type of GNN that performs localized convolution over document node features and pregiven links using Graph Laplacian.

• **GAT [37]:** Graph Attention Network, a GCN extension which integrates attention mechanism into the neighborhood aggregation.

• **Text GCN [43]:** a method that applies GCN to text classification. It creates word-word links and document-word links, ignores their types and merges them into a homogeneous network, and applies GCN to learn node representations.

• **BERT [21]:** Deep Bidirectional Transformers for Language Understanding. We include this special baseline to show the state-of-the-art performance that can be achieved if we do not explicitly consider the connections between documents (but make full use of the sequential information within each text).

5.1.2 ***Datasets***.

We evaluate our method and baselines on six widely used benchmarks whose statistics are summarized in Table 1: Hep-th [18] is a collection of high-energy physics papers with citations; Wiki [41] is a collection of wiki web pages with hyperlinks; CoraEnrich [8] and Citeseer [42] are two collections of machine learning papers with citations; PubMed [42] is a collection of bio-medical papers with citations; Reuters-8 [43] is a subset of Reuters 21578 news reports.

Although having the raw text as input is appreciated for making more accurate predictions, notice they are not necessarily required by both our method and most of our baselines, which only require the word-count matrix as the text input. Being able to work with datasets with discrete text attributes is also important in practice. Examples include classifying people's social media profiles using tagging information, and classifying long articles based on their extracted keywords. CiteSeer, PubMed, and Wiki belong to such cases, for which we use only 1-gram (word) as term nodes in our method. Also notice that the last dataset Reuters-8, does not have any pre-given links. It is a popular dataset for link-free text classification and we further include it to demonstrate our method's potential of leveraging the advantage of collective classification to tackle a link-free text classification problem.

5.1.3 ***Preprocessing and Training Configurations***.

For all datasets that come with raw text, we preprocess the text data following a standard pipeline: we tokenize the text as [43], remove English stopwords defined by NLTK [1], and remove low-frequency words appearing less than 5 times in the corpus.

Dataset	Docs	Links	Vocab.	Classes	Type
Hep-th [18]	11752	134,956	21,614	4	papers
CoraEnrich [8]	2,708	5,278	25,935	7	papers
Wiki [41]	2,405	17,981	4,973	19	web pages
CiteSeer [42]	3,327	4,552	3,703	6	papers
PubMed [42]	19,717	44,324	500	3	papers
Reuters-8 [43]	7,647	-	7,688	20	news report

Table 1: Summary of the dataset statistics.

To tune our model, we adopt grid search over three hyperparameters: number of topics $|T|$, number of terms $|W|$, and dimensions of hidden layers. We report their search ranges and values for other hyperparameters such as learning rate and dropout in Appendix. The train-val-test split for most datasets are 60%-20%-20%, except for Hep-th and Reuter-8 where we use pregiven public split. We sufficiently train our model using Adam optimizer until the loss converges, and then use the validation set to select the best epoch checkpoint for testing. For all baselines' tuning, we follow their default hyperparameter (search) settings reported in their paper or implementation. We ran all the experiments 10 times using different random seeds and report the mean Accuracy ± 95% confidence interval as the metric, following [8, 41, 43].

5.2 Results and Discussion

Table 2 presents the test accuracies of each model. We can see that the two variants of our proposed method significantly outperform all the state-of-the-art baselines on most datasets (p<0.05 by student t-test). On average, our best model variant improves over the accuracy of the strongest baseline by 1.5% in relative. We additionally have the following observations:

• The class of GNN methods (GNN, GAT, Text GCN) are the strongest baselines overall. While our proposed method also adopts a graph neural network base, it still outperforms the GNN methods by a stable margin. We attribute this performance gain to our usage of latent textual links: general GNNs do not take advantage of any text-specific structural information; in contrast, our method exploits the advantage of text by using the very rich latent textual links among the nodes.

• Text GCN's performance lies in between GAT and our method. However, it still consistently underperforms our method. The main reason is that Text GCN depends only on shared words to bridge two documents, which in practice is usually noisy. It also ignores different relationship types in the corpus when doing propagation. In comparison, our topic-associated links reveal more and cleaner structural information of the corpus, and our GEAT better uses the type information of relationships.

• CANE, PTE, and Text GCN can not work with discrete text attributes (*e.g.* Wiki, CiteSeer, PubMed). This limits their generalizability to tasks where only discrete text tags are available, such as classifying people's social media profiles using some text tags, or classifying long articles based on their extracted keywords. However, our method well copes with these scenarios because it does

[1]http://www.nltk.org

Model	Hep-th	CoraEnrich	Wiki	CiteSeer	PubMed	Reuters-8
TF-IDF	42.04 ± 0.92	73.80 ± 2.45	71.71 ± 3.59	74.27 ± 1.44	84.28 ± 0.54	93.74 ± 0.43
LDA [2]	42.25 ± 0.69	74.02 ± 0.77	78.24 ± 0.62	76.24 ± 1.18	86.59 ± 0.75	94.48 ± 0.50
BERT [6]	45.16 ± 0.84	78.97 ± 2.43	-	-	-	97.11 ± 0.38†
DeepWalk [28]	39.83 ± 0.61	81.44 ± 1.63	69.91 ± 2.90	58.51 ± 2.23	86.92 ± 0.74	-
TADW [41]	44.55 ± 0.70	87.29 ± 0.58	80.72 ± 0.82†	74.67 ± 0.63	86.82 ± 0.61	-
paper2vec [8]	33.26 ± 2.98	80.58 ± 0.90	70.01 ± 2.62	59.26 ± 3.12	62.56 ± 1.33	-
CANE [36]	42.88 ± 0.87	82.84 ± 1.09	-	-	-	-
PTE [33]	44.34 ± 0.51	88.50 ± 1.28	-	-	-	96.46 ± 0.16
GCN [17]	45.49 ± 0.38	87.63 ± 0.66	77.96 ± 0.45	76.68 ± 0.53†	87.15 ± 0.38†	-
GAT [37]	45.88 ± 0.49	87.71 ± 0.69	78.12 ± 0.76	76.21 ± 0.59	86.94 ± 0.37	-
Text GCN [43]	46.01 ± 0.30†	90.02 ± 0.39†	-	-	-	97.07 ± 0.10
GEAT-LDA-pt	**48.23 ± 0.45***	**90.42 ± 0.37***	**81.22 ± 0.63**	**77.04 ± 0.56**	**88.48 ± 0.58***	97.11 ± 0.15*
GEAT-VTG	**48.36 ± 0.31***	**90.55 ± 0.46***	**80.99 ± 0.42**	**77.26 ± 0.33***	**88.59 ± 0.69***	**97.25 ± 0.10***

Table 2: Performance in Accuracy (mean in percentage ± 95% confidence interval). † highlights the best baselines. *, bold font, bold font* respectively highlights the case where our models' performance exceeds the best baseline on average, by 70% confidence, by 95% confidence.

not require word occurrence (though if available it can still make use of word occurrence via n-grams).

• Comparing the two variants, GEAT-LDA-pt and GEAT-VTG, we see that the latter performs slightly better. The reason is that compared to GEAT-LDA-pt, GEAT-VTG allows more freedom in constructing the network – GEAT-VTG's topic-associated link weights are trained from scratch with supervised signals, while GEAT-LDA-pt fine-tunes the link weights set by a pretrained topic model. The topic model is pretrained in an unsupervised way. Therefore, GEAT-VTG is likely to generate link weights better optimized for the classification task.

• BERT can achieve high performance without any inter-document relationships, because it well utilizes the sequential information within each document. In comparison, our method makes prominent progress in another direction by exploiting inter-document relationships, which also proves to be a great success.

5.3 Ablation Studies

A central goal of our experiment is to validate the effectiveness of each building component in our heterogeneous text graph. Therefore, we carried out a series of ablation studies to check how removing or replacing each component affects the performance. The ablations and results are summarized in Table 2 and here is their introduction: Ab. 1 uses our unmodified model of GEAT-LDA-pt as the comparison base; Ab. 2-6 remove different subsets of links we use: Ab. 2 removes the given non-textual document links E_{DD}; Ab. 3 removes the document-word links E_{DW}; Ab. 4 removes all the topic-associated links which include document-topic links E_{DT} and topic-term links E_{TW}; Ab. 5 removes all textual links, E_{DT}, E_{TW} and E_{DW}; Ab. 6 removes all types of links. In Ab. 7, we initialize the term nodes differently: instead of looking them up from an embedding table trained from scratch, we try using the pretrained Glove word embeddings over our corpus to initialize the terms. That is, each term (n-gram) is initialized using the average of all its words' pretrained embeddings. Finally, in Ab. 8 we try making every n-gram in the corpus into a term node, so there is no aggressive pre-filtering step which only leaves less than 10% of them.

Table 3 presents us several interesting insights into our model via row-wise comparisons:

•**Ab. 1 vs. Ab. 2-6:** these comparisons show that our manually created textual links help significantly. In fact, every type of link in our constructed heterogeneous network contributes to the performance, which further demonstrates the importance of mining and leveraging rich inter-document relationships for classification.

•**Ab. 6 vs. Ab. 2, 5**: we see that the proper usage of textual links not only significantly improves performance, but the improvement can be even larger than that of using non-textual links alone (on Hep-th dataset, for example).

•**Ab.1 vs. Ab. 7**: an interesting debate over our framework's design is whether one should use pretrained word embeddings as the initial term node features. Comparisons between these two ablation studies show that using pretrained word embeddings actually slightly hurts the performance. This interesting phenomenon may be because the pretrained word embeddings are trained in a fully unsupervised manner. Therefore, they may distribute in a sub-optimal feature space which is hard to change even with supervised fine-tuning afterwards. In comparison, our term embeddings are trained with supervised signals from scratch, so they may capture semantics that are better contextualized and customized for the classification task.

•**Ab. 1 vs. Ab. 8**: this comparison shows us that using the full vocabulary set to build the text network can be a less desirable choice. The interpretation is that we may only need a small number of vocabularies to define cleaner and more reliable textual links for a corpus, and to build a well-connected text network out of it to do classification. We include more discussions on this in Sec. 5.4.

5.4 Hyperparameter Sensitivity

We analyze the effect of two important hyperparameters: the number of topic nodes $|T|$, and the number of term nodes $|W|$. In each experiment, we set a target hyperparameter to different values, and for each value find its corresponding optimal performances by grid-searching a best combination of other hyperparameters. The

No.	Ablation	Cora Enrich	Hep-th	Reuters-8
1	original method: GEAT-LDA-pt	90.42 ± 0.37	48.23 ± 0.45	97.11 ± 0.15*
2	original method - all non-textual links (E_{DD})	84.87 ± 0.52	46.53 ± 0.47	97.11 ± 0.15*
3	original method - document-word textual links (E_{DW})	88.91 ± 0.39	46.83 ± 0.39	96.30 ± 0.26
4	original method - topic-associated textual links (E_{DT}, E_{TW})	87.52 ± 0.41	46.56 ± 0.35	96.52 ± 0.23
5	original method - all textual links (E_{DW}, E_{DT}, E_{TW})	87.07 ± 0.45	45.93 ± 0.42	93.76 ± 0.31†
6	original method - all links ($E_{DW}, E_{DT}, E_{TW}, E_{DD}$)	73.80 ± 1.49	42.11 ± 0.55	93.76 ± 0.31†
7	original method + pretrained word embeddings	90.43 ± 0.56	48.01 ± 0.13	97.02 ± 0.10
8	original method + use all vocabulary	82.79 ± 0.64	46.32 ± 0.21	96.84 ± 0.16

Table 3: Ablation studies with GEAT-LDA-pt, performance in Accuracy (mean in percentage ± 95% confidence interval) reported. ∗†: the results are pairwise the same, because Reuters-8 does not have any non-textual links.

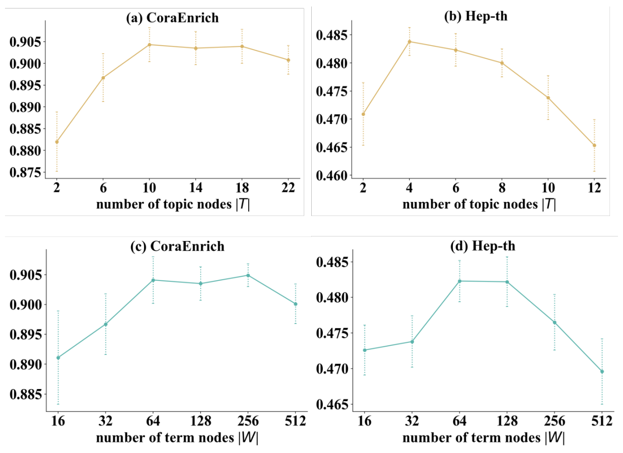

Figure 2: Test Accuracies w.r.t different hyperparameters.

results are plot in Fig. 2. Plots (a), (b) demonstrate the importance topic nodes: $|T|$ has a significant effect on the performance, and in particular a too small $|T|$ will likely hurt the performance. Also, the performance peaks with certain optimal range of $|T|$, which is around 1 ∼ 1.5 times the class number of the dataset (the class numbers are listed in Table 1). From Plots (c), (d) we further observe that similar to $|T|$'s case, the performance also peaks at a certain range of $|W|$. This optimal range, however, is way smaller that the magnitude of vocabulary size (only accounting for around 1% in fact). This suggests that we may not need learn the embeddings of all the words / terms like PTE [33] in order to classify a document. This is consistent with Ab. 8 in Table 3, which shows that learning embeddings for all vocabulary actually hurts performance a lot.

5.5 Visualizing Topic-term Attentions

We visualize the topic-term attentions learned by the first layer of our **GEAT-VTG** model. We set up the exploration as the following: we first train our **GEAT-VTG** model on CoraEnrich dataset by setting the number of topic nodes equal to the number of label classes, 7. Then, we retrieve the topic-term attention weights from the first layer of our model. For each topic node, we rank the top 8 terms linked to the topic node with the highest attention weight. Table 4 lists the ranking for 4 topic nodes. We can see from the term examples that the attention weights learned by our model turn out to be highly predictive of the category already, though we do not directly use them to make predictions. We attribute this interesting characteristic to the success of our VTG strategy and its good integration with GEAT's edge attention mechanism: the VTG strategy learns good topic-term link attributes and GEAT further uses those attributes to compute edge attention. If we instead adopt the node-node attention mechanism used by GAT [37] , it would be less likely to see the strong predictive power of attentions since their attention mechanism tends to ignore the link attributes.

Topic Node 1	Topic Node 2	Topic Node 3	Topic Node 4
logic	theorem	tree	action
infer	bound	decision	reinforce
clause	query	attribute	case base
belief	polynomial	classifier	simulate
language	finite	decision tree	instruct
causal	learnable	machine learning	goal
decision	learning algorithm	classifier	game
logic program	converge	learning algorithm	policy

Table 4: Top-8 terms with highest attention weights connected to different topic nodes. The attentions weights are extracted from the first layer of GEAT-VTG trained on CoraEnrich dataset.

6 CONCLUSION

In this work, we study the collective text classification problem by exploiting different latent *textual links* among documents. We investigate an optimal way of defining and leveraging those relationships via the construction of a well-connected heterogeneous text network. We pair the network construction process with two strategies to generate topic-associated link weights, and further presents a Graph Edge-Attention Network to model the complex textual relationships and to learn predictive document embeddings. Extensive experiments show that our proposed method significantly outperforms various state-of-the-art baselines by a large margin. Ablation studies further validate the effectiveness of various components in our method.

ACKNOWLEDGMENTS

We thank

REFERENCES

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv:2004.05150* (2020).
[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[3] Vitor Carvalho and William Cohen. 2006. Improving "email speech acts" analysis via n-gram selection. In *Proceedings of the Analyzing Conversations in Text and Speech*. 35–41.
[4] William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, Vol. 161175. Citeseer.
[5] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391–407.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[7] Yajuan Duan, Furu Wei, Ming Zhou, and Heung-Yeung Shum. 2012. Graph-based collective classification for tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 2323–2326.
[8] Soumyajit Ganguly and Vikram Pudi. 2017. Paper2vec: Combining graph and text information for scientific paper representation. In *European Conference on Information Retrieval*. Springer, 383–395.
[9] Chaobo He, Hanchao Li, Xiang Fei, Atiao Yang, Yong Tang, and Jia Zhu. 2017. A topic community-based method for friend recommendation in large-scale online social networks. *Concurrency and Computation: Practice and Experience* 6, 29 (2017), n–a.
[10] Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 50–57.
[11] John Houvardas and Efstathios Stamatatos. 2006. N-gram feature selection for authorship identification. In *International conference on artificial intelligence: Methodology, systems, and applications*. Springer, 77–86.
[12] David Jensen, Jennifer Neville, and Brian Gallagher. 2004. Why Collective Inference Improves Relational Classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Seattle, WA, USA) *(KDD '04)*. Association for Computing Machinery, New York, NY, USA, 593–598. https://doi.org/10.1145/1014052.1014125
[13] Chanwoo Jeong, Sion Jang, Hyuna Shin, Eunjeong Park, and Sungchul Choi. 2019. A context-aware citation recommendation model with BERT and graph convolutional networks. *arXiv preprint arXiv:1903.06464* (2019).
[14] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[15] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615* (2015).
[16] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl
[18] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.
[19] Chang Li and Dan Goldwasser. 2019. Encoding social information with graph convolutional networks forPolitical perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2594–2604.
[20] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* (2016).
[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[22] Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *Advances in Neural Information Processing Systems*. 8208–8217.
[23] Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering Discrete Latent Topics with Neural Variational Inference. In *International Conference on Machine Learning*. 2410–2419.
[24] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning*. 1727–1736.
[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[26] Nishith Pathak, Colin DeLong, Arindam Banerjee, and Kendrick Erickson. 2008. Social topic models for community extraction. In *The 2nd SNA-KDD workshop*, Vol. 8. 2008.
[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.
[29] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. New Jersey, USA, 133–142.
[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
[31] Marc A Smith, Lee Rainie, Ben Shneiderman, and Itai Himelboim. 2014. Mapping Twitter topic networks: From polarized crowds to community clusters. *Pew Research Center* 20 (2014), 1–56.
[32] Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488* (2017).
[33] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: Predictive Text Embedding through Large-Scale Heterogeneous Text Networks *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 1165–1174. https://doi.org/10.1145/2783258.2783307
[34] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.
[35] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical Dirichlet processes. In *Advances in neural information processing systems*. 1385–1392.
[36] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1722–1731.
[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[38] Rui Wang, Deyu Zhou, and Yulan He. 2019. Atm: Adversarial-neural topic model. *Information Processing & Management* 56, 6 (2019), 102098.
[39] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*. 606–615.
[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).
[41] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information.. In *IJCAI*, Vol. 2015. 2111–2117.
[42] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.
[43] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7370–7377.

A ADDITIONAL EXPERIMENTAL SETUPS

A.1 Tuning Our Model

The hyperparameters search range of our method is reported in Table 5. For each dataset, we first heuristically look for a combination learning rate and dropout rate within the range reported, fix them, and then conduct comprehensive grid search over other hyperparameters for the best set of values.

Hyperparameter	Search Range / Value
learning rate	1e-4, 2e-4, 4e-4
dropout rate	0, 0.1, 0.3, 0.5
hidden dimension h	50, 100, 150, 200
λ_D (for GEAT-VTG only)	0.2, 0.5, 1
number of topic nodes $\lvert T\rvert$	$0.5C$, C, $1.2C$, $1.5C$, $2C$
number of term nodes $\lvert W\rvert$	32, 64, 128, 256, 512, 1024

Table 5: Hyperparameter search range for the two variants of our method. C is the class number of each dataset.

A.2 Tuning Baselines

BERT We experiment with RoBERTa [21], the state-of-the-art version of BERT [6]. We use the official implementation and pretrained weights of RoBERTa provided by the HuggingFace's transformer library [40]. Following common practices, we replace the last linear layer and fine-tune RoBERTa with the text classification task. Following the default hyper-parameter setting, the batch size is set to 16 while the learning rate is set to 2×10^{-5}. The maximum input sequence length of RoBERTa is 512 tokens. We also experiment with LongFormer [1], which supports longer input sequences by optimizing the memory usage of RoBERTa's self-attention mechanism. However, we did not observe accuracy improvements compared to RoBERTa. On each dataset, we fine-tune RoBERTa for 10 epochs, and we select the best model using the validation set.

PTE We use the official implementation provided by PTE [33]. PTE learns document classification in two phases: In the feature learning phase, PTE learns document representation using the training data. We then extract the document features for the training set, validation set, and test set. In the classification phase, we use the one-vs-rest (OvR) logistic regression model in the scikit-learn package. Following the default hyper-parameter setting, we set the window size in the convolution layer as 3, the number of feature maps as 100, and the dimension of word vectors as 100.

Text GCN We use the official implementation provided by Text GCN [43], and use most of their default hyperparameter settings: learning rate = 0.02, #epochs = 200, dropout rate = 0.5, with early stopping that stops the training if the validation performance does not increase for more than 10 epochs.

CANE We use the official implementation provided by CANE [36], and use the configurations mentioned in their "config.py" file: #epochs = 200, batch size = 64, hidden dimension = 200, learning rate = 1e-3.

GCN and GAT For the two baselines, we adopt their PyTorch Geometric [2] implementations, and tune their learning rate, dropout, and hidden dimensions by the same search range reported in Table 5. Both methods share the same preprocessing step with our proposed method.

TADW We use the official MatLab implementation provided by [41]. We follow most of their default hyperparameter settings: $\lambda = 0.2$, $k = 80$, and word vector dimension = 200.

paper2vec Unfortunately, the author does not release its code and so we implemented the paper ourselves. The only hyperparameter to be tuned is the dimension k and we follow the papers advice to search it in 160, 180, 200.

[2]https://pytorch-geometric.readthedocs.io/