Weakly Supervised Concept Map Generation through Task-Guided Graph Translation

Jiaying Lu, Xiangjue Dong, and Carl Yang

Abstract—Recent years have witnessed the rapid development of concept map generation techniques due to their advantages in providing well-structured summarization of knowledge from free texts. Traditional unsupervised methods do not generate task-oriented concept maps, whereas deep generative models require large amounts of training data. In this work, we present *GT-D2G* (Graph Translation-based Document To Graph), an automatic concept map generation framework that leverages generalized NLP pipelines to derive semantic-rich initial graphs, and translates them into more concise structures under the weak supervision of downstream task labels. The concept maps generated by *GT-D2G* can provide interpretable summarization of structured knowledge for the input texts, which are demonstrated through human evaluation and case studies on three real-world corpora. Further experiments on the downstream task of document classification show that *GT-D2G* beats other concept map generation methods. Moreover, we specifically validate the labeling efficiency of *GT-D2G* in the label-efficient learning setting and the flexibility of generated graph sizes in controlled hyper-parameter studies.

Index Terms—Concept Map Generation, Graph Translation, Weak Supervision, Document Summarization, Document Classification.

1 Introduction

STANDING out for the clear and concise structured knowledge representation, concept maps have been widely applied in knowledge management [1], [2], document summarization [3], [4], information retrieval [5] and educational science [6], [7]. Fig. 1 shows toy examples of concept maps derived from a document describing *"Moon Landing"*, where nodes in the graph indicate important concepts and links reflect interactions among concepts. Although concept maps are helpful in both providing interpretable representations of texts and boosting the performance of downstream tasks, the creation of concept maps is challenging and time-consuming.

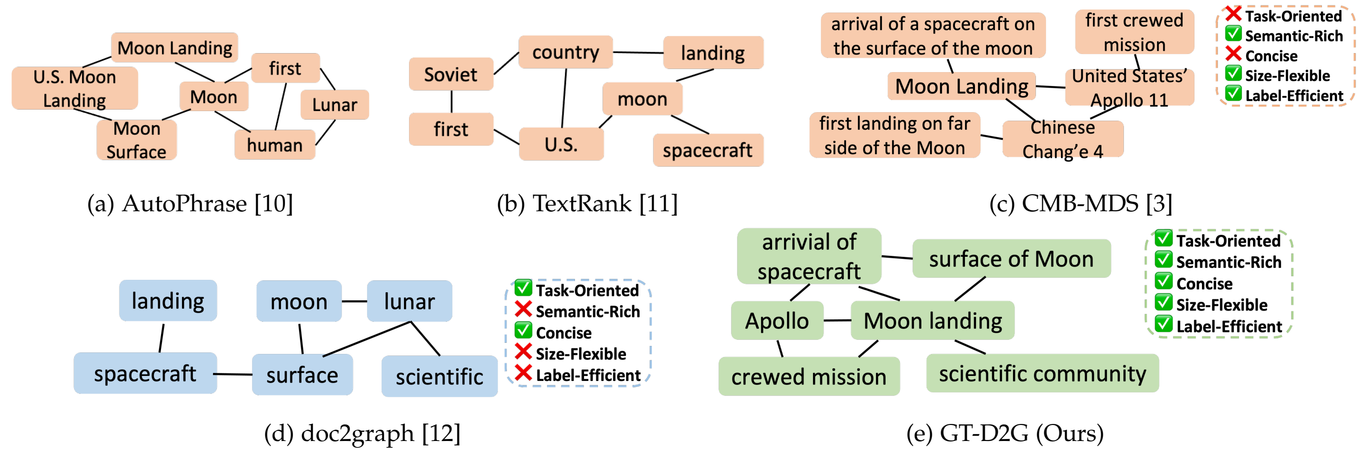
Traditionally, concept map generation follows a multi-step pipeline including concept extraction, relation identification and graph assembling [3], [8], [9], where auxiliary resources and carefully designed heuristics are often required. However, the separation of concept map construction and downstream tasks easily deviates the generated graphs from what the real task needs. For example, Figures 1a, 1b, 1c provide examples of concept maps constructed from such unsupervised ad hoc processes. Although the sample document has the label of *science*, the extracted concepts of "U.S. Moon Landing" (1a), "Soviet" (1b) and "Chinese Chang'e 4" (1c) are more related to the label of *politics*. As a consequence, these deviating concepts will likely degrade the performance of document classification. Moreover, nodes chosen by these traditional methods often lack conciseness due to their heavy reliance on ad hoc pipelines. For instance, in Fig. 1a, the concept map contains redundant concepts such as "Moon" and "Moon Surface" as concepts mined by *AutoPhrase* are mainly based on frequency features; while in Fig. 1c, the concepts are rather verbose due to the OpenIE component for concept generation in *CMB-MDS*.

On the other hand, research efforts have been made to automatically generate concept maps from documents under the weak supervision from text-related downstream tasks. *Doc2graph* [12] is one pioneering study that achieves this goal through a fully end-to-end neural network model. However, due to the lack of linguistic analysis, the generated concepts often suffer from semantic incompleteness and the links between concepts are often noisy. For example in Fig. 1d, one compound concept *"moon landing"* is preferable than two separated concepts *"landing"* and *"moon"* as the former carries more precise and complete semantic information. Moreover, while the weakly supervised training diagram enables *doc2graph* to generate concept maps at scale, we observe the downside of being not label-efficient. In other words, *doc2graph* is sensitive to training signals and it requires a significant amount of weak supervision to construct meaningful concept maps, as discussed in §5.4. Finally, the size of concept maps generated by *doc2graph* is fixed due to its rigid technical design, while the ideal size of graphs should vary according to the complexity of documents being represented.

Inspired by both existing methods, we propose a graph translation-based neural concept map generation framework that simultaneously leverages existing NLP pipelines and receives weak supervision from downstream tasks, dubbed as **GT-D2G** (Graph Translation-based Document To Graph). The integration of NLP pipelines effectively assists *GT-D2G* to address the semantic incompleteness issue of *doc2graph* by introducing both words and phrases as concept candidates. Meanwhile, the initial semantic-rich graphs constructed by the NLP pipeline bring in *a priori* knowledge from the linguistic side, thus alleviating the label

• *Jiaying Lu and Carl Yang are with the Department of Computer Science, Emory Univeristy, Atlanta GA, 30322; Xiangjue Dong is with the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas, 77843.*
E-mail: jiaying.lu@emory.edu, xj.dong@tamu.edu, j.carlyang@emory.edu

Manuscript received Nov, 2021; revised Dec, 2022.



(a) AutoPhrase [10] (b) TextRank [11] (c) CMB-MDS [3]

(d) doc2graph [12] (e) GT-D2G (Ours)

Fig. 1: Toy examples of concept maps on the topic *"Moon Landing"* generated by different methods.

inefficient issue of *doc2graph*. In *GT-D2G*, concepts and their interactions are generated iteratively through a sequence of nodes and adjacency vectors, which ensures deeper coupling between nodes and links for more meaningful results and resolves the fixed size issue of *doc2graph*. On the other hand, guided by the weak supervision from downstream tasks, *GT-D2G* is also able to generate task-oriented concept maps that provide preferable support to specific downstream tasks, while eliminating the redundancy issue of traditional unsupervised methods, specifically through the incorporation of a penalty over content coverage. To sum up, concept maps generated by our proposed *GT-D2G* method are task-oriented, semantic-rich, concise, size-flexible, and label-efficient, as illustrated in Fig. 1e.

The overall technical design of *GT-D2G* bridges the gap between the NLP pipeline-driven concept map generation and the end-to-end neural concept map generation by presenting a task-guided graph translation neural network. Specifically, our *GT-D2G* framework consists of several sub-modules: Initial Graph Constructor, Graph Encoder, Graph Translator, and Graph Predictor. In particular, the input text is first processed by an NLP pipeline-based Initial Graph Constructor to obtain a set of concept candidates with their associated relations. Then, a graph pointer network [13] based Graph Translator equipped with a graph convolution network [14] based Graph Encoder is applied upon the initial concept map to simultaneously select important concepts and links. A graph isomorphism network [15] based Graph Predictor is finally responsible for predicting the downstream task labels from the translated graph. The whole model is trained by weak signals from downstream tasks, while also regularized by a deliberately designed penalty term towards graph conciseness. As a result, *GT-D2G* provides high-quality concept maps that are both effective for downstream tasks and interpretable towards knowledge management.

In this work, an extensive suite of experiments has been conducted on text corpora from three domains: news, scientific papers, and customer reviews. Through experiments on the downstream task of document classification, we demonstrate that the proposed *GT-D2G* framework outperforms both traditional concept map generation baselines and the state-of-the-art neural method *doc2graph*, while a comprehensive ablation study shows the effectiveness of each of our novel designs. The quality and interpretability of generated graphs are supported by rigorous human evaluation and rich case studies. Finally, we specifically validate the labeling efficiency of *GT-D2G* in the label-efficient learning settings and the flexibility of generated graph sizes in controlled hyper-parameter studies.

2 Related Work

2.1 Automatic Concept Map Generation

The concept map generation task is first introduced by [16], where the task definition and a benchmark dataset *EDUC* are proposed. In [16], a corpus of 30 document clusters in which each contains around 40 source documents and 1 crowdsourcing summary concept map was provided. A keyphrase-based approach concept map generation approach was also proposed and evaluated in terms of precision, recall, and F1 of concept propositions (concept pairs). We do not include this approach as its follow-up study proposes a more advanced model. *CMB-MDS* [3], the extended model from the same research group, adapted the task definition, and then proposed an approach that utilized coreference resolution module to merge coreferent concepts and integer linear programming module to globally optimize the summary concept maps. Different from above mentioned studies, *doc2graph* [12] did not rely on human-generated concept maps as training samples. Instead, the graph generation model in doc2graph was trained with weak supervision from downstream tasks.

Side-by-side comparison between *doc2graph* and *GT-D2G*. Both *doc2graph* and *GT-D2G* models are capable of generating concept maps without ground-truth training samples. However, *GT-D2G* is not a simple extension of *doc2graph*. As shown in Fig. 1, *GT-D2G* owns task-oriented, semantic-rich, concise, size-flexible, and label-efficient properties while *doc2graph* only owns two of them. The semantic-rich property is achieved by our proposed NLP pipeline constructed graph. The size-flexible property is achieved by the RNN-based edge decoder and the kernel-based length regularizer. The label-efficient property is achieved by the graph translator. Regarding the overall model designs, *GT-D2G* is

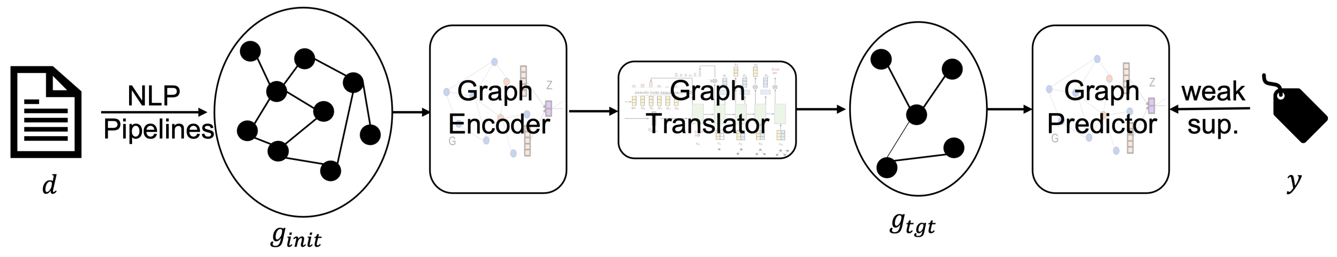


Fig. 2: Overview of proposed *GT-D2G* framework.

based on the graph translation framework while *doc2graph* is based on generating graphs from scratch. In experiments, this work conducts additional extensive human evaluation, quantitative downstream application evaluation, labeling efficiency evaluation, flexibility evaluation, and comprehensive case studies, while *doc2graph* only conducts downstream application evaluation and simple case studies.

2.2 Graphical Structures for Texts

Graph structures have been extensively used in various NLP and text mining tasks, including keyword-based graph [10], [11] and parsing-based graph [17], [18], [19]. *TextRank* [11] builds a word graph to represent the text, and connected words using a fixed-size sliding window. The word importance score is then calculated by a modified PageRank formula. *TextRank* has been widely applied in keyword extraction and sentence extraction, and here we utilize it as a baseline for concept map generation. However, concepts are restricted as words and the number of concepts to keep cannot be inferred by the algorithm itself. *AutoPhrase* [10] is another popular method for keyword extraction and keyphrase mining tasks. Top-k high-quality phrases can be extracted as concepts and then a graph can be constructed using concepts co-occurring in the same sentences. Although *AutoPhrase* can generate both words and phrases as concepts, the drawback is that the size of the graphs has to be fixed. In this work, we incorporate dependency parsing [19] to provide the semantic-rich initial graph and propose a graph translation model to enable the size-flexible concept map generation.

2.3 Graph Generation and Translation Methods

The focus of this paper, concept map generation, is a subtask of the graph generation task. GraphVAE [20] and GraphRNN [21] are two pioneering works that learn to a distribution model $p_{model}(\mathbb{G})$ over a set of observed graphs $\mathbb{G} = \{G_1, \ldots, G_n\}$. Following this line, several flow-based models (GraphAF [22], GraphDF [23], etc.) and diffusion models (GRAND [24], EDM [25], etc.) are proposed for graph generation. However, these generative models all require a significant amount of ground-truth graph samples, while in concept map generation task these samples are often scarce. CondGen [26] explores generating novel graphs conditioned on unseen semantic labels by explicitly modeling the relationships between graph context and structures. *doc2graph* [12] further tackles the graph generation under a weakly supervised manner. On the other hand, our proposed *GT-D2G* follows a graph translation (GT) paradigm to utilize the semantic-rich graph derived from NLP pipelines. VJTNN [27] proposes a VAE-based model for molecular graph translation. GT-GAN [28] proposes a GAN-based model for directed weighted graph translation. Similarly, these models require high-quality paired datasets. SegTran [29] is a semi-supervised autoencoder-based GT model that requires only a small fraction of training data. The basic idea of SegTran is to use a specific encoder/decoder for the source/target graph, and translate the graph in the latent domain. Our proposed *GT-D2G* further addresses the lack of training paired graph issue, and it is capable of generating concept maps from free-form texts using weak supervision from downstream tasks.

3 Problems Statement

Problem Definition. We focus on the novel problem of weakly supervised concept map generation. It can be defined as follows: Given a text corpus $\mathcal{D}_l = (d_1, \ldots, d_{i_l})$ with corresponding labels $\mathcal{Y} = (y_1, \ldots, y_{i_l})$ of certain downstream text-related tasks, we aim at generating concept maps $g_i = \{\mathcal{C}_i, \mathcal{M}_i\}$ for each document $d_i \in D_u$ where D_u is a set of unlabeled documents. As can be seen from the definition, there are no ground-truth concept maps paired with the input text. Instead, weak or distant supervision from downstream tasks is provided. The downstream text-related tasks are very flexible, possibly ranging from document classification, retrieval, ranking, relation inference, *etc*. The major output is concept maps $\mathcal{G}$ for all documents $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$. A document $d \in \mathcal{D}$ is indeed a sequence of words, *i.e.*, $d_i = (w_{i,1}, \ldots, w_{i,|d_i|})$. A concept map $g_i = \{\mathcal{C}_i, \mathcal{M}_i\}$ is an undirected graph which focuses on the concepts $\mathcal{C}_i$ and their interactions $\mathcal{M}_i$ in the span of d_i. $\mathcal{C}_i = (c_{i,1}, \ldots, c_{i,|c_i|})$ is a set of n concepts that can be words, phrases, or sentence fragments depending on the downstream tasks, and $\mathcal{M}_i \subseteq \mathbb{R}^{n\times n}$ indicates the interaction strength (*i.e.*, edge weight) among concepts in $\mathcal{C}_i$. Moreover, the auxiliary output is the predicted labels $\hat{\mathcal{Y}}$ for unlabeled documents $\mathcal{D}_u$.

We summarize the essential challenges of weakly supervised concept map generation as two folds in the following. **Challenge 1.** *How to construct concept maps without training samples?* The concept map is a natural structure for representing interactions of concepts introduced in a document. However, despite its promising utility, the usage of concept maps so far is still limited. One critical reason is the lack of available training samples to drive the data-eager graph generative models. Therefore, models should be able to efficiently generate without training samples.

Remarks 1. Unsupervised concept map generation models have made important progress toward automatically generating large-scale concept maps from text data. Concept maps would be more helpful if generated concepts and interactions are relevant to downstream tasks. Considering the downstream task labels come with the text corpus, it is important that models can explicitly utilize these signals to guide the concept map generation process.

Challenge 2. *How to guarantee the quality of generated concept maps?* Quality is another important property if concept maps aim to thrive in successful applications. Ideally, high-quality concept maps should concisely distill and represent the key information in the text. On the other hand, the downstream task performance can partially reflect the quality of generated concept maps.

Remarks 2. Human evaluation should be regarded as the major examination criteria for the quality of generated concept maps, since no ground-truth concept maps are available in the datasets. Multiple metrics that cover different quality aspects are proposed and described in §5.2. Moreover, automatic metrics $f(\mathcal{Y}_u, \hat{\mathcal{Y}}_u)$ (*e.g.*, accuracy, MRR, *etc.*) can be included as another indicator for quality evaluation.

4 Proposed Approach

Fig. 2 gives an overview of the proposed *GT-D2G* (Graph Translation based Document-To-Graph) framework: A proper NLP pipeline is used to extract salient phrases from document d and construct the initial semantic-rich concept map g_{init}. A Graph Encoder then encodes each node of g_{init} into a node-level embedding $\boldsymbol{Q_i}$, and also represents the whole g_{init} as a dense vector by aggregating all its node embeddings. A Graph Translator is responsible to identify the nodes needed to be kept in the target graph g_{tgt} as well as proposing links among kept nodes iteratively. Once the nodes and links are generated, the target graph g_{tgt} is fed into a Graph Predictor to produce a document label $\hat{y}$, which can be trained towards the ground-truth label y. The whole encoder-translator-predictor neural network is thus weakly supervised by the classification signal in an end-to-end fashion. In the following subsections, we expand with more technical details.

4.1 Enriching Concept Maps with Semantics

As we motivated before, one major drawback of doc2graph [12] is that single words are directly picked from the raw texts through a Pointer Network [13] and considered as nodes in the final concept map. However, words purely picked by a simple Pointer Network can easily be of low-quality [30]. Moreover, phrases are often preferable to represent concepts, especially noun phrases as semantically complete concepts [10]. For instance, extracting two nodes “deep”, “learning” from a computer science paper is incomplete while “deep learning” as one concept node is semantically more meaningful and accurate. Some researchers propose to concatenate words that occur adjacently in the input document as extracted phrases to solve this issue, although potential heuristic post-processing is needed. In *GT-D2G*, we aim to enrich concept maps with semantics by leveraging existing NLP pipelines [19]. For simplicity and generalization concerns, we intentionally choose the most popular yet reliable NLP tools for initial concept map construction, which can be further extended according to application scenarios.

Node Generation. To avoid complicated pre-processing, we use multiple classic NLP tools in *GT-D2G* to extract noun phrases, verb phrases, and adjectives as node candidates in the initial concept map. Sentence segmentation, pos-tagging, lemmatization, and constituency parsing are conducted for every document. Since constituency parsing detects sub-phrases of given sentences, we then first extract basic noun phrases from constituency parsing results. The basic noun phrases extraction algorithm is deterministic so that any noun phrase not containing other noun phrases is considered valid. After all basic noun phrases are identified, verb phrases and adjectives remaining in the text are extracted. Other discourse units such as adverbs and prepositions are discarded since they typically do not contain much knowledge or information. Due to the fact that multiple words can refer to the same concept, determinants such as “a”, “an”, “the” are removed from the node mentions, and words are replaced by their lemmas. Moreover, pronouns need to be merged into coreferent mentions to obtain a clean initial concept map. Thus, the coreference resolution technique is used to resolve all pronoun expressions in documents. We use the popular Stanford CoreNLP [31] for all steps mentioned above.

Link Generation. For links between extracted nodes, we follow the sliding window idea introduced in keyphrase extraction studies [11]. Nodes that occur within a fix-sized sliding window are connected to each other. Therefore, the initial concept maps are undirected graphs $g_{init} = \{C_{init}, M_{init}\}$. The link construction module is flexible in *GT-D2G* so that any algorithms can be applied to construct weighted links or directed links. For instance, we can directly use the whole parsing tree or filter out certain types of relations for link generation. The graph ensemble process is trivial once nodes and links are extracted.

4.2 Task Guided Graph Translation

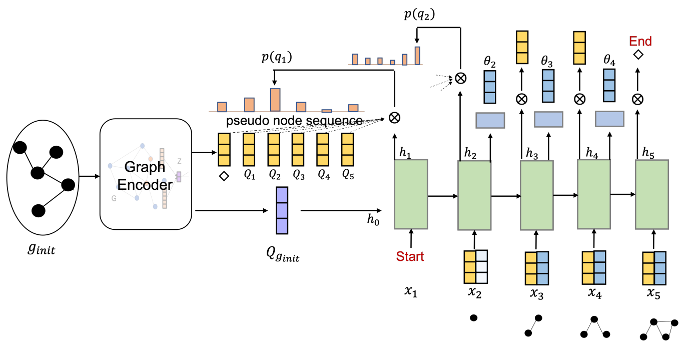


Fig. 3: Graph Translator. Green rectangles denote RNN cells that take the previous time step chosen node q_{t-1} and generated adjacency vector $\boldsymbol{\theta_{t-1}}$ as input. The RNN state vector $\boldsymbol{h_t}$ is updated at every time step, and is initialized by the graph level representation of initial graph $\boldsymbol{Q_{g_{init}}}$.

Graph Encoder. Before graph translation, the model has to first learn to understand the initial graph. For this purpose, we adopt the recent successful graph representation learning model, *i.e.*, Graph Convolutional Network (GCN) [14] as our Encoder. The node embeddings $\boldsymbol{Q}^{(k)}$ are learned after the k-th layer of GCN by the following equation

$$\boldsymbol{Q}^{(k)} = \mathrm{ReLU}(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{M}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{Q}^{(k-1)}\boldsymbol{W}_{\boldsymbol{Q}}^{(k)}), \qquad (1)$$

where $\boldsymbol{W}_{\boldsymbol{Q}}^{(k)}$ is learnable parameters in the k-th layer of GCN, $\tilde{\boldsymbol{M}} \subseteq \mathbb{R}^{n\times n}$ is the adjacency matrix $\boldsymbol{M}_{init}$ with additional self-connections and $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{M}}_{ij}$ is the diagonal degree matrix. The input node embeddings $\boldsymbol{Q}^{(0)}$ are the concatenations of phrase embeddings, normalized frequency feature, and normalized location feature. The phrase embedding of each node is the average of pre-trained word embeddings (in practice, we use GloVe [32]). The frequency feature and the location feature reflect the importance of the concept in the original text and are normalized by min-max scaling per graph. Besides the node-level embeddings, we also compute the graph-level embedding as $\boldsymbol{Q}_{\boldsymbol{g}_{init}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{Q}_{\boldsymbol{i}}^{\boldsymbol{k}}$ to encode the global contextual information in the initial graph.

Graph Translator. Our graph translator aims to choose the most informative nodes that are also beneficial to downstream tasks from the initial graph, while proposing links among the chosen nodes accordingly. In particular, the Graph Translator generates a sequence of nodes and their corresponding adjacency vectors based on the initial concept map g_{init}– to be specific, its node-level embeddings $\boldsymbol{Q}_{\boldsymbol{i}}$ ($i \in [1, n]$) and graph-level embedding $\boldsymbol{Q}_{\boldsymbol{g}_{init}}$ produced by the Graph Encoder. Since we expect to preserve the semantic rich and task-relevant concepts in the initial graph and only pick out a subset of nodes, we adopt the Pointer Network [13] from keyword selection and novelly extend it into a graph version to generate a sequence of pointers for the selection of the most important nodes from the initial concept map. After each node is selected, we get inspiration from GraphRNN [21] to also generate its corresponding adjacency vector which contains links to previously selected nodes. However, the original GraphRNN only works on the transductive learning setting when there is an actual graph as input to learning from. Therefore, we need to make several novel modifications to GraphRNN before seamlessly integrating it into our Graph Pointer Network (GPT) towards our novel setting of task-guided graph translation.

Graph Pointer Network. Since the original Pointer Network [13] works on sequential text data, we convert the non-sequential nodes in the initial concept map into a pseudo node sequence according to positions of node mentions in the source document, illustrated as the yellow bars in Fig. 3. The order of pseudo node sequence is flexible and can be replaced with any other order for proper reasons (e.g., node degree order). Here we just follow the most intuitive way and do not observe significant performance differences when using other orders. In our GPT, we use a one-directional RNN decoder to model the process of translating a sequence of nodes and links from an initial graph, denoted as the green rectangles in Fig. 3. In practice, we choose GRU [33] as the implementation. In order to start the translation from the whole initial graph, the hidden state of the RNN decoder is initialized by $\boldsymbol{h}_{\boldsymbol{0}} = \boldsymbol{Q}_{\boldsymbol{g}_{init}}$, and the input of the first step is $\boldsymbol{x}_{\boldsymbol{1}} = (0, \ldots, 0)^{\intercal}$. Therefore, the hidden state that encodes the "graph translation state" is updated by

$$\boldsymbol{h}_{\boldsymbol{t}} = \mathrm{RNN}(\boldsymbol{x}_{\boldsymbol{t}}, \boldsymbol{h}_{\boldsymbol{t-1}}), \qquad (2)$$

where $\boldsymbol{h}_{\boldsymbol{t-1}}$ denotes the hidden state from the last time step, and $\boldsymbol{x}_{\boldsymbol{t}}$ denotes the input at the current time step. More specifically, we compute $\boldsymbol{x}_{\boldsymbol{t}}$ as the representations of both nodes and links generated from the last time step, which can be denoted as

$$\boldsymbol{x}_{\boldsymbol{t}} = [\boldsymbol{q}_{\boldsymbol{t-1}}; \boldsymbol{\theta}_{\boldsymbol{t-1}}], \qquad (3)$$

where $[\cdot;\cdot]$ denotes vector concatenation. $\boldsymbol{q}_{\boldsymbol{t-1}} = \boldsymbol{Q}_{\boldsymbol{i}}$ is the node embedding from the Graph Encoder of the last selected node i, and we defer the explanation towards adjacency vector $\boldsymbol{\theta}_{\boldsymbol{t-1}}$ to the later part of this subsection.

Deeply coupled node and link generation. Once we obtain the RNN decoder hidden state $\boldsymbol{h}_{\boldsymbol{t}}$, the node selection process can be described by the following equations

$$e_{i,t} = \boldsymbol{v}^{\intercal}\tanh(\boldsymbol{W}[\boldsymbol{Q}_{\boldsymbol{i}}; \boldsymbol{h}_{\boldsymbol{t}}]), \qquad (4)$$

$$\mathsf{p}_{t,i} = \mathsf{p}(\boldsymbol{q}_{\boldsymbol{t}} = \boldsymbol{Q}_{\boldsymbol{i}}) = \frac{\exp(e_{i,t})}{\sum_{j=1}^{n}\exp(e_{j,t})}, \qquad (5)$$

where $\boldsymbol{v} \subseteq \mathbb{R}^{d_e}$ and $\boldsymbol{W} \subseteq \mathbb{R}^{d_h \times d_e}$ are learnable parameters for calculating the unnormalized node selection score $e_{\cdot,t}$ at time step t for every node in initial graph node set C_{init}. Our GPT then selects the i-th node with the maximum score by

$$i = \arg\max_{i}(\mathsf{p}_{t,i}), \qquad (6)$$

adds the selected node into the translated target graph and feeds $\boldsymbol{q}_{\boldsymbol{t}} = \boldsymbol{Q}_{\boldsymbol{i}}$ into the RNN decoder at the next time step. To improve the semantic completeness of selected concept nodes, we also adapt the coverage loss in [34], by maintaining a coverage vector $\boldsymbol{c}_{\boldsymbol{t}} = \sum_{t'=0}^{t-1}\mathsf{p}(\boldsymbol{q}_{\boldsymbol{t'}})$ that accumulates the generated attention so far, while adding the following loss to enforce the model to pay more attention to nodes not covered yet:

$$L_{cov} = \sum_{d_i \in \mathcal{D}}\sum_{t_j \in d_i}\min(\mathsf{p}(\boldsymbol{q}_{\boldsymbol{t'}}), \boldsymbol{c}_{\boldsymbol{t}_j}). \qquad (7)$$

To deeply couple the generation process of nodes and links so that the target graph (*i.e.*, final concept map) is meaningful, we get inspired by the recent deep graph generation model of GraphRNN [21]. Specifically, in our GPT, at each time step, after a new node is generated, we immediately generate its associated adjacency vector regarding all links between it and all previously generated nodes, as denoted by smaller blue rectangles in Fig. 3 and described in the following equation

$$\boldsymbol{\theta}_{\boldsymbol{t}} = f_{out}(\boldsymbol{h}_{\boldsymbol{t}}), \qquad (8)$$

where $\boldsymbol{\theta}_{\boldsymbol{t}}$ is the length $t-1$ adjacency vector for the chosen node at time step t that is output by f_{out}. Based on slightly different goals for link generation, we design two variants of f_{out}: the *path* variant and the *neigh* variant. The former models the adjacency vector generation as generating a path connecting some previously picked nodes to the currently picked one, focusing on the higher-order sequential information among concepts. Hence, f_{out}^{path} is implemented as another RNN that connects to the hidden state of the RNN decoder. On the other hand, the *neigh* variant interprets the generation problem as generating all possible neighbors of the currently picked node from all previously picked nodes, focusing on the first-order neighborhood structures of concepts. Therefore, f_{out}^{neigh} is implemented as a multi-layer perceptron (MLP) with non-linear activation. The weights of f_{out} are shared across all time steps to reduce the number of parameters and alleviate overfitting. In our experiments, we find the *neigh* variant to be preferable over the *path* variant, which can be intuitively attributed to the fact that structural information is more important than sequential information among concepts.

Graph Predictor. After generating a sequence of nodes $q_1, \ldots, q_T$ and adjacency vectors $\theta_1, \ldots, \theta_T$, we assemble the target graph as

$$g_{tgt} = \{C_{tgt}, M_{tgt}\} = \{(q_1, \ldots, q_T), (\theta_1, \ldots, \theta_T)\}. \quad (9)$$

For the downstream graph-level prediction, we adopt a *Graph Isomorphism Network* (GIN) [15] due to GIN's superior discriminative power to capture different graph structures. More specific, we adopt the sum operator as the neighborhood aggregation function, and an MLP as the center node and neighbor nodes combination function:

$$\boldsymbol{q^{(k)}} = \mathrm{ReLU}((\boldsymbol{M_{tgt}} + (1 + \epsilon^{(k)})\boldsymbol{I})\boldsymbol{q^{(k-1)}}\boldsymbol{W_q^{(k)}}), \quad (10)$$

where $\boldsymbol{M_{tgt}} \subseteq \mathbb{R}^{T\times T}$ is the adjacency matrix of translated concept map, $\boldsymbol{I} \subseteq \mathbb{R}^{T\times T}$ is a identity matrix (*i.e.*, self connection), and $\epsilon^{(k)}$, $\boldsymbol{W_q^{(k)}}$ are learnable parameters for GIN's k-th layer. Furthermore, the graph label (*i.e.*, document category in our case) $\hat{y}$ is obtained by an additional two-layer MLP on the graph representation:

$$\hat{y} = \mathrm{MLP}(\mathrm{concat}(\mathrm{sum}(\boldsymbol{q^{(k)}})|k = 1, \ldots, K)), \quad (11)$$

where the graph representation is achieved by summing all node embeddings from the same layer, and then concatenating summed embeddings across all layers.

4.3 Training Techniques

The whole model is trained in a weakly supervised end-to-end fashion, by computing the cross-entropy loss for the downstream task– document classification as we focus on in this work, and the coverage loss for the node selection in our GPT. Specifically, we have

$$L_{cls} = -\sum_{d_i \in \mathcal{D}} \mathsf{p}(\hat{y_i}) \log \mathsf{p}(y_i), \quad (12)$$

$$L = L_{cls} + \lambda * L_{cov}, \quad (13)$$

where λ is a tunable hyper-parameter.

One technical challenge exists for the node selection operation that selects the node with maximum pointer attention $i = \arg\max_i(\mathsf{p}(\boldsymbol{q_t} = \boldsymbol{Q_i}))$ during the graph translation process in GPT. Firstly, the max value selection operation implemented as *argmax* is non-differentiable, thus leading to the lost gradient after node selection. Secondly, *argmax* is a deterministic sampling operator, thus making the GPT loses exploration ability. The exploration ability or stochastic sampling is important during the early training stages of GPT, because the predicted probability to select a node is not very reliable at that time. Inspired by the re-parameterization tricks for categorical variables sampling [35], [36], [37], we adopt a hard-version *Gumbel-Softmax* to sample one-hot vectors from the predicted probabilities, so that the node selection process in GPT is differentiable and stochastic. The sampled probability $P_{t,i}$ to choose node i at time step t then becomes:

$$P_{t,i} = \mathrm{softmax}(\log(\mathsf{p}_{t,i}) + G_i, \tau), \quad (14)$$

where $\mathsf{p}_{t,i}$ is the predicted probability as defined in Eq (5), $G_i \sim \mathrm{Gumbel}(0, 1)$ is the i-th random variable sampled from the Gumbel distribution, and τ is the temperature parameter for *softmax*. We set a relative large temperature to enforce $P_t = (P_{t,1}, P_{t,2}, \ldots, P_{t,n})$ has the one-hot vector shape. During training, we use $P_t \cdot Q$ to represent selecting one particular node for gradient backpropagation.

Moreover, to generate concept maps of flexible sizes, we incorporate the special *"EOS"* node at the first position of pseudo node sequence, denoted as "◇" in Fig. 3. The end of an output node sequence is determined when the *"EOS"* is predicted. For the completeness of concept maps, we penalize node sequences that are too short, which can be implemented by applying a penalty to *"EOS"* node predicted at every time step as follows

$$L_{len} = \sum_{d_i \in \mathcal{D}} \sum_{t_j \in d_i} \mathrm{Penalty}(t_j) \cdot \mathsf{p}(\boldsymbol{q_{t_j}} = \text{“}EOS\text{”}). \quad (15)$$

The function $\mathrm{Penalty}(t) > 0$ defines a penalty curve depending on the current time step t. In our implementation, we choose the RBF kernel function $\Phi(t, t') = exp(-\frac{\|t-t'\|^2}{2\sigma^2})$ for the penalty curve [38]. Therefore, the overall loss function for *GT-D2G* is:

$$L = L_{cls} + \lambda_1 * L_{cov} + \lambda_2 * L_{len}. \quad (16)$$

To sum up, our whole framework is trained in an end-to-end fashion, while Graph Encoder, Graph Translator, and Graph Predictor are guided by the downstream task with the goal of reducing classification loss. In this way, each module is jointly learned and enhanced. Moreover, the translation process is regularized by the coverage loss and graph size loss, aiming to produce high-quality concept maps depending on the input documents' characteristics.

4.4 Complexity Analysis

To analyze the computational efficiency of the proposed model, we present the *GT-D2G* training algorithm for one input initial concept map (one input document). The actual implementation is based on mini-batch training, and is publicly available[1]. For obtaining graph representation of the initial concept map (L3-L4), the time complexity is $\mathcal{O}(Knd^2 + Kmd)$, where K is the number of GCN

1. *GT-D2G*: https://github.com/lujiaying/GT-doc2graph

Algorithm 1: *GT-D2G* Training Algorithm

```
Data: initial concept map $g_{init} = \{\mathcal{C}_{init}, \mathcal{M}_{init}\}$,
      input node embeddings $\{Q_v^0, \forall v \in \mathcal{C}_{init}\}$,
      ground truth graph label $y$
Result: translated concept map $g_{tgt} = \{\mathcal{C}_{tgt}, \mathcal{M}_{tgt}\}$,
        predicted graph label $\hat{y}$
1  Initialize GT-D2G parameters;
2  while not converge do
       /* Obtain graph representation of $g_{init}$ */
3      Update node embedding
       $\boldsymbol{Q^{(k)}} = \text{GCN}_{\text{Enc}}(\boldsymbol{Q^{(0)}}; k)$ by Eq. 1;
4      Update graph embedding
       $\boldsymbol{Q_{g_{init}}} = \text{pooling}(\{\boldsymbol{Q_v^{(k)}}, \forall v \in \mathcal{C}_{init}\})$;
       /* Translate $g_{init}$ into $g_{tgt}$ step-by-step */
5      while not generate "EOS" node do
6          Prepare Graph Translator (RNN) input
           $(\boldsymbol{x_t}, \boldsymbol{h_{t-1}})$ by initilization or previous step
           results;
7          Update hidden state of Graph Translator $\boldsymbol{h_t}$
           by Eq. (2);
8          Generate node $q_t$ by Eq. (4), (5), (6);
9          Generate adjacency vector $\theta_t$ by Eq (8);
       /* Predict graph label */
10     Assemble the translated concept map $g_{tgt}$ by
       Eq. (9);
11     Predict the graph label $\hat{y}$ by Eq. 10, 11;
       /* Backpropagate the weak supervision */
12     Compute the overall loss $L$ by Eq. (16);
13     Update model parameters with the gradients of
       $L$.
```

encoder layers, d is the embedding dimensions (128 in all layers), n is the number of nodes in g_{init} (tens of nodes in our experiments), m is the number of edges in g_{init} (*e.g.*, close to one hundred edges in our experiments). The time complexity can be further simplified into $\mathcal{O}(Knd^2)$ since $nd \gg m$. For graph translation (L5-L9), the time complexity is $\mathcal{O}(TKnd^2)$, where T is the size of the translated concept map, K is reused to represent the number of RNN decoder layers (*e.g.*, we set both GCN encoder, RNN decoder and GIN classifier layer sizes as 2), d is reused to represent the RNN embedding dimensions (*e.g.*, we set the hidden dimension to 128 for all modules). For the graph label prediction(L10-L11), the time complexity is $\mathcal{O}(KTd^2)$ which is similar to GCN encoder analysis. Therefore, the overall time complexity for proposed *GT-D2G* is $\mathcal{O}(Knd^2 + TKnd^2 + KTd^2) = \mathcal{O}(TKnd^2)$.

It is worth noting that the construction of initial concept maps is quite efficient, as the toolkit we employed (*e.g.*, JVM-based Stanford CoreNLP [31]) mainly utilize pre-trained models or rule-based annotators for the NLP pipelines. Moreover, *doc2graph*'s time complexity is $\mathcal{O}(TK\|\mathcal{D}\|d^2)$, where $\|\mathcal{D}\|$ denotes the number of words of input document. *GT-D2G* is more efficient than *doc2graph*, due to the fact that $\|\mathcal{D}\| \geq n$ in most cases. However, the advantage of *doc2graph* is that it does not require NLP pipelines to derive the initial concept maps.

5 Experiments

In this section, we evaluate our proposed *GT-D2G* framework focusing on the following four research questions:
RQ1: How is the quality of *GT-D2G* generated graphs?
RQ2: How do *GT-D2G* and its variants perform in comparison to other document classification methods?
RQ3: Is *GT-D2G* label efficient?
RQ4: Can *GT-D2G* generate flexible sizes of concept maps?

5.1 Experiment Settings

Datasets. Our experiments are conducted on three real-world text corpora [12]: *NYT*, *AMiner*, and *Yelp*. Different from [12], for the *Yelp* dataset, we re-grouped the 1-5 star reviews into negative, neutral and positive ratings. The statistics of the three datasets are listed in Table 1. For standard document classification, we follow the setting in [12] to randomly split the labeled documents into 80% for training, 10% for validation, and 10% for testing. We choose accuracy as the metric for document classification tasks. To get a stable result, we run each model three times and report the mean ± standard deviation.

TABLE 1: Statistics of three datasets.

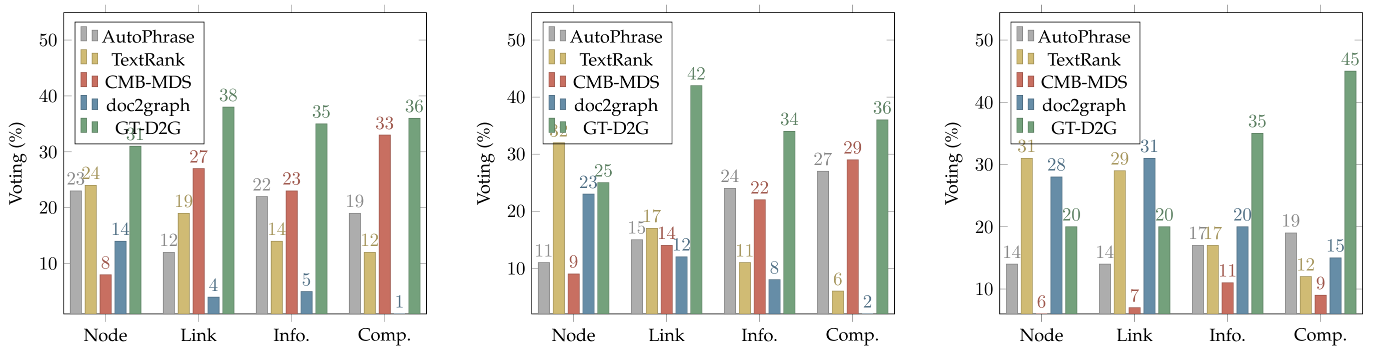
Dataset	#doc	#word	#category	Init Concept Map		
				#node	#edge	#degree
NYT	13,081	88.64	5	34	84	4.9
Aminer	21,688	87.27	6	34	81	4.8
Yelp	25,357	71.59	3	28	76	5.4

Compared Methods. We compare *GT-D2G* with two sets of baselines described as follows:
Graph-Based Methods as major competitors.

- ***AutoPhrase*** [10]: This is a Pos-Guided Phrasal Segmentation model for phrase mining. We use the top-n highest quality phrases mined from input text as concepts and connect concepts in same sentence. The edge weights is computed as $w_{ij} = 1 - e^{-c_{ij}}$, where c_{ij} denotes sentence-level co-occurring times of concept i and j.
- ***TextRank*** [11]: A word co-occurrence graph is first constructed using a sliding window that connects any two words within the window. We use words with top-n maximum PageRank values as concepts. The edge weights are computed in the same way as *AutoPhrase*.
- ***CMB-MDS*** [3]: We use its pipeline to construct concept map and filter out concepts with low importance scores to keep top-n concepts. The edge weights are set to 1 according to the *CMB-MDS* implementation.
- ***doc2graph*** [12]: *doc2graph* is a neural concept map generation model that is capable of generating concept maps through distant document classification supervision. We follow their implementation to pre-define graph size as n.

Text-Based Methods as performance benchmarks.

- ***Bi-LSTM*** [39]: *Bi-LSTM* is a commonly used RNN model in text classification that learns the long-term dependencies in the document. We train *Bi-LSTM* on the training set using the output from last time-step to predict document categories.



(a) NYT (b) AMiner (c) Yelp

Fig. 4: Human evaluation results on (a) NYT, (b)AMiner, (c)Yelp based on four proposed metrics.

- ***BERT-base*** [40]: *BERT* has achieved excellent performance on a wide range of NLP tasks as a state-of-the-art language model. In our experiment, We fine-tune the pre-trained *BERT-base* model on the classification task.

Implementation Details. We implement *GT-D2G* using Pytorch [41] and DGL [42], with code publicly available[1]. Implementations of the compared baselines are either from open-source project (*BERT*[2]) or the original authors (*Bi-LSTM*/ *AutoPharse*/ *TextRank*/ *CMB-MDS*/ *doc2graph*[3]). We optimize *GT-D2G* through the Adam optimizer with learning rate to $3e-4$ and max epoch to 500. The temperature parameter τ for Gumbel-softmax starts from a big number (*e.g.* 3 or 5) and then anneals along with training epochs to encourage exploration on the later stage. To get a higher accuracy, we set batch size to 64 for training. The hidden layer dimension of GCN, RNN and MLP are set to 128, and the number of GNN layers in all GCN, GIN models are 2. For RBF kernel function used to penalize overlength node sequence, σ and t_{prime} are set to 4 and 0, respectively. We choose GRU for RNN used in generating nodes and edges for simplicity sake. All other hyper-parameters are tuned separately on the validation set.

5.2 Human Evaluation (*RQ1*)

TABLE 2: Correlation coefficients among the five peer annotators with manual responsiveness scores on a total of 300 documents of NYT, AMiner, Yelp (100 each).

Peer Scoring	Node	Link	Info.	Comp.
NYT	0.50	0.89	0.57	0.67
AMiner	0.76	0.80	0.75	0.93
Yelp	0.73	0.79	0.70	0.92

Human evaluation is critical to answer *RQ1*, *i.e.* evaluating the quality of generated concept maps, since there are no ground-truth concept maps on the three document classification datasets. Five expert annotators are hired to evaluate graphs generated from the text data by five methods: *AutoPhrase*, *TextRank*, *CMB-MDS*, *doc2graph*, and *GT-D2G*. More specifically, on each dataset, we randomly sample 100 document with associated graphs of each method. or each document, annotators are asked to rank the five concept maps in terms of four metrics:

Node: regardless of downstream tasks, whether nodes are semantic complete, in proper length and not redundant.
Link: whether links between nodes are consistent with the text and make sense.
Informativeness: whether the generated graph is helpful for the downstream task.
Completeness: whether the generated graph covers the most salient information of the original text from different aspects.

Correlation Coefficient is a widely used indicator to estimate the inter-annotator agreement (ITA). However, we observe that explicitly annotating the rank among all five concept maps leads to low inter-annotator agreement. Therefore, we allow annotators to pick k ($k \leq 3$) graphs for each metric as top graphs, as long as they think these k graphs are of the same best quality. That means, if an annotator thinks two graphs by *doc2graph* and *GT-D2G* are competitive in Informativeness, she can mark both two as top graphs without distinguishing which is the best. The top max-k graph annotation guideline gives high Correlation Coefficient scores, as can be seen in Table 2.

The human evaluation results are shown in Fig. 4. The value on y-axis indicates the percentage of the data that the annotator think the method performs best under the corresponding metric. For the metrics of *Informativeness* and *Completeness*, annotators reached a high degree of consistency that our approach *GT-D2G* outperforms other baseline methods significantly. Moreover, *GT-D2G* performs best on *NYT* for *Node* metrics and *NYT* and *AMiner* for *Link* metrics.

Case Studies. The concept maps constructed by five methods are shown in Fig. 5 and 8. In general, *AutoPhrase* can represent meaningful concepts using phrases, but sometimes prone to generate duplicate nodes (*e.g.*, two *"mobile device"* in *AMiner* example). *TextRank* select meaningful concepts in word-level which are beneficial for the downstream tasks (*e.g.*, *"beethoven"* in *NYT*, *"mobile"* in *AMiner*, and *"amazing"* in *Yelp*), but the links among the selected concepts are not consistent with the original text. The nodes generated from *CMB-MDS* usually contain abundant information but are often in sentence-level, which are not concise and redundant. *doc2graph* can generate useful concepts with meaningful links, however, the nodes are mainly word-level (*e.g.*,

2. *BERT*: https://github.com/huggingface/transformers
3. *doc2graph*: https://github.com/JieyuZ2/doc2graph

NYT (Arts): PLAINFIELD, Mass. — On a recent sunny afternoon, Matt Haimovitz entered a carpentry workshop here that doubles as a music studio and gently pulled the door shut. The garden of the 19th-century farmhouse echoed with the shouts of children ... but Mr. Haimovitz cupped his hand around its neck with loving pride: "This is my Beethoven cello."

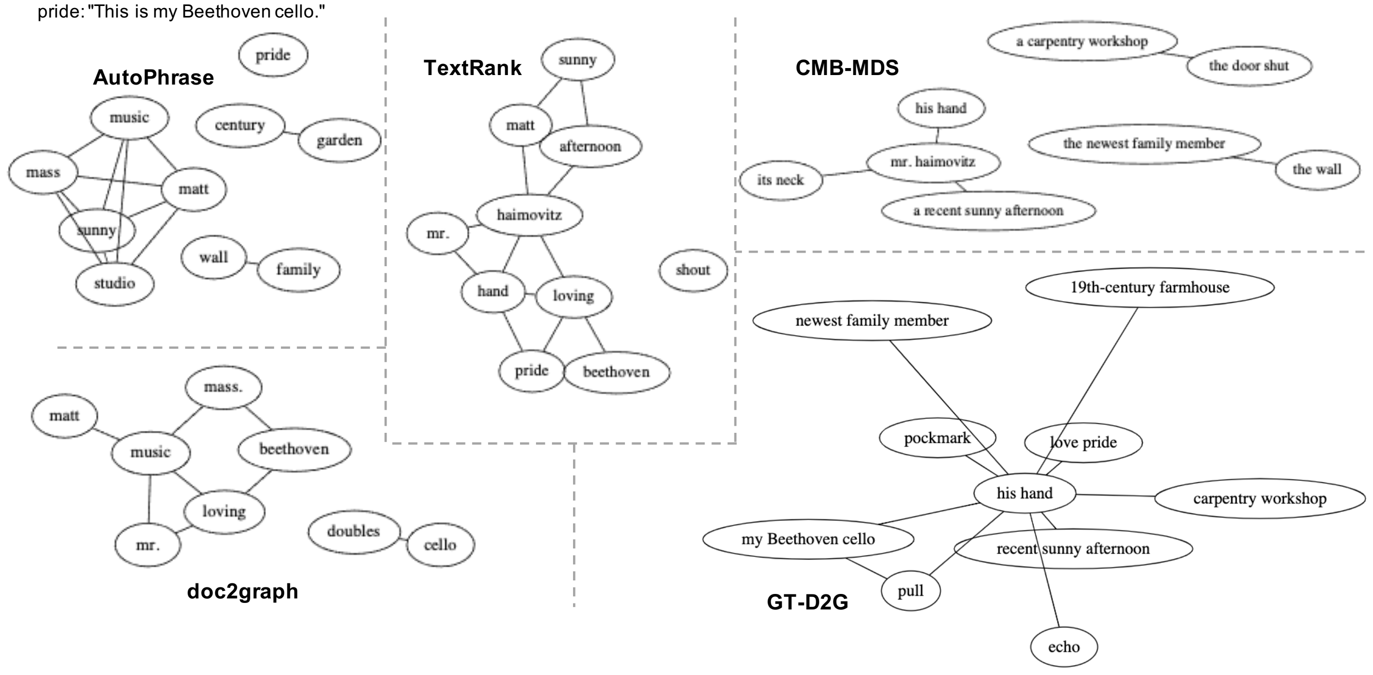


Fig. 5: Concept maps generated by various models for case studies.

"mr." instead of *"mr. haimovitz"* in *NYT*) and sometimes contain *"<unk>"* or *"-"* which indicate the limitation of this method. Our approach, *GT-D2G* can represent concepts in both word-level and phrase-level ways which are concise, semantic-rich, and beneficial for downstream tasks (*e.g.*, *"beethoven cello"* in *NYT*).

5.3 Classification Results (*RQ2*)

TABLE 3: Document classification accuracies(%).

Model	NYT	AMiner	Yelp
Bi-LSTM	87.52 ± 3.01	59.32 ± 2.71	78.46 ± 1.46
BERT-base	$\underline{97.54} \pm 0.16$	$\underline{73.62} \pm 0.06$	$\underline{85.34} \pm 0.08$
AutoPhrase	92.42 ± 0.65	59.63 ± 0.85	72.66 ± 0.33
TextRank	89.48 ± 0.07	57.47 ± 0.31	70.25 ± 0.61
CMB-MDS	87.68 ± 0.72	51.93 ± 2.02	65.63 ± 2.07
doc2graph	90.81 ± 1.00	67.06 ± 1.32	79.89 ± 0.52
GT-D2G-init	93.65 ± 0.86	66.76 ± 1.77	80.15 ± 0.80
GT-D2G-path	95.26 ± 0.13	68.23 ± 0.23	80.86 ± 0.97
GT-D2G-neigh	95.34 ± 0.33	$\mathbf{68.53} \pm 1.02$	80.92 ± 0.50
GT-D2G-var	$\mathbf{95.46} \pm 0.49$	68.37 ± 1.05	$\mathbf{80.98} \pm 0.51$

To answer *RQ2*, we conduct the document classification experiments on three text corpora. The generated concept maps have n concepts. To compare our methods with baseline methods conveniently, we set $n = 10$ for all graph-based baselines and non-flexible *GT-D2G* variants (*-path* and *-neigh*). For *GT-D2G-init*, n is equal to the total number of nodes of constructed initial graphs. For the flexible *GT-D2G* variant (*-var*), we set $n \leq 10$. Table 3 shows the classification performance of our methods and the compared methods. We observe that *GT-D2G* consistently outperforms all baseline methods except *BERT-base* on all three datasets, which indicates that the integration of semantic-rich initial concept maps from NLP pipelines and graph translation based on the weak supervision in our methods benefit the downstream tasks significantly. Notably, both *Bi-LSTM* and *BERT-base* are not capable of generating concept maps. As we mentioned before, the goal of *GT-D2G* is not to beat all SOTA document classification methods, but to achieve a competitive performance while providing interpretable structured knowledge representation. Consequently, in the following comparison elaborations, we exclude these two methods when we mention "baseline methods".

Compared with traditional graph-based approaches, *GT-D2G* gains 3%, 15%, 11% over the best results of traditional approaches on *NYT*, *AMiner*, and *Yelp*, respectively. Moreover, it surpasses the end-to-end *doc2graph* method by 5%, 2% and 1%, correspondingly. As mentioned in the toy example (Fig. 1) and §5.1, both *AutoPhrase*, *TextRank* and *CMB-MDS* are existing unsupervised concept map generation models. These three models are capable of generating concept maps according to their own customized metrics (*e.g.*, frequency-based, connectivity-based, summarization-based), but they can not utilize the downstream task's signals to supervise the generation process. Consequently, concepts generated by these models are not task-oriented, and thus leading to the poor classification performance. On the other hand, *doc2graph* is the only compared model that is specifically designed for weakly-supervised concept map generation. As reflected in the experimental results, *doc2graph* is the major competitor of our *GT-D2G* (excluding the SOTA document classification models).

To better understand the effectiveness of our proposed techniques (§4), we closely study the four variants of *GT-D2G* regarding the effectiveness of NLP pipelines (*-init*), node-and-link iterative generation (*-path* and *-neigh*), and

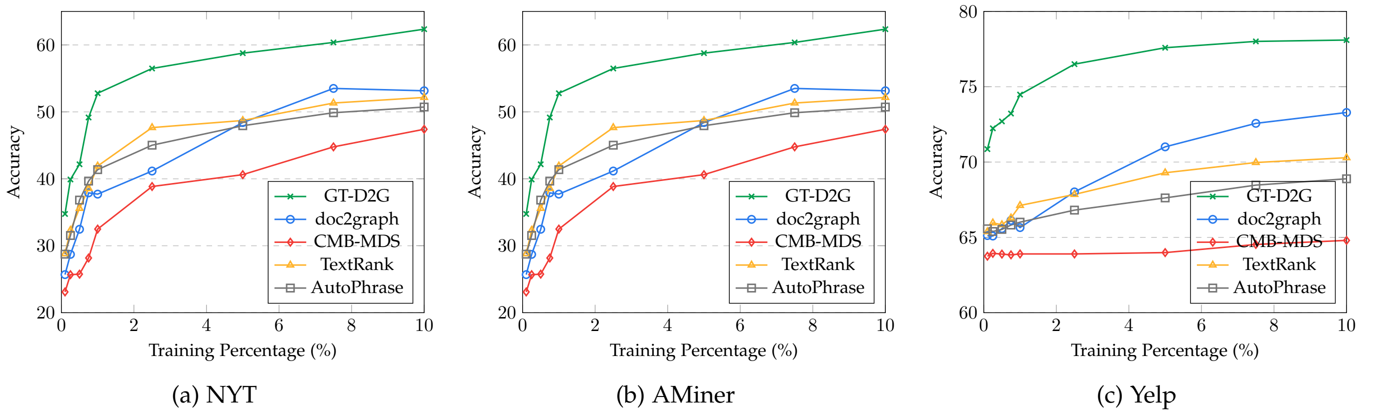


Fig. 6: Test accuracies by varying the proportions of training data (ranging from 0.1%, 0.25%, . . . to 10.00%).

flexible-size graph generation (*-var*). In particular, to evaluate the effectiveness of incorporating NLP pipelines, we implement *GT-D2G-init* that directly encodes all nodes in the initial semantic-rich concept maps to make predictions. Table 3 show that *GT-D2G-init* outperforms all traditional graph-based baselines with 1.23 on *NYT*, 7.13% on *AMiner*, and 7.49% on *Yelp*. Comparing *GT-D2G-init* with *doc2graph*, *GT-D2G-init* achieves 1.23% and 0.26% gains on *NYT* and *Yelp*, while *GT-D2G-init* is worse by 0.3% on *AMiner*. Hence, the observed experimental results support the benefits of utilizing concept maps derived from NLP pipelines. Upon *GT-D2G-init*, the other three variants add the Graph Translator module to obtain a more concise concept map, since the initial concept maps often contain 20-40 nodes and the translated concept maps contain less than 10 nodes. According to the experimental results, the translated concept maps are preferable to initial concept maps, as they can further improve *GT-D2G-init* by 1.81% on *NYT*, 1.77% on *AMiner*, and 0.83% on *Yelp*.

To explore a proper way to generate edges, we implement and compare two methods, *GT-D2G-path* and *GT-D2G-neigh*. *GT-D2G-path* only generates edges based on the relations of concepts in text sequence while *GT-D2G-neigh* links each node with its all possible neighbors. As shown in Table 3, *GT-D2G-neigh* is consistently better than *GT-D2G-path* on all three datasets, which well supports our argument that generating edges among all possible neighbors is preferable to generating edges as a sequence of paths starting from the node. Furthermore, *GT-D2G-var* addresses the fixed size issue of *doc2graph* and the experiment results of *GT-D2G-var* illustrate the benefits of generating flexible size of concept maps. More discussion about generating size-flexible concept maps are in §5.5.

5.4 Labeling Efficiency Evaluation (*RQ3*)

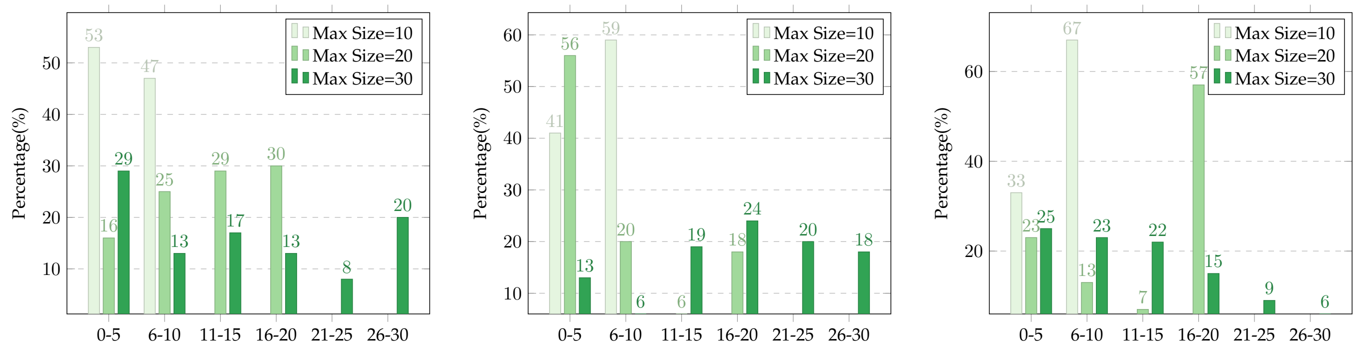
To demonstrate the labeling efficiency of *GT-D2G* over other concept map generation methods, we conduct experiments with different proportions (0.1%, 0.25%, 0.50%, 0.75%, 1.00%, 2.50%, 5.00%, 7.50%, 10.00%) of the training data. To get a stable test accuracy, we take the average value among three trials of each experiment by applying different random seeds. The average test accuracies of *NYTimes*, *AMiner*, and *Yelp* datasets were shown in Fig. 6 respectively, which answer *RQ3*.

We can observe that our approach *GT-D2G* has higher test accuracy than the other approaches from the beginning, with only 0.1% of the training data. In addition, with the increasing of the training data size, our model has steeper growth curves of test accuracy, which shows its effectiveness in exploiting limited supervision, and makes it maintain excellent performance during the whole label efficiency evaluation with limited labeled data. These results demonstrate the labeling efficiency of our model, which is enabled by the semantic-rich initial concept maps (§4.1) and the Gumbel-softmax training technique (§4.3). Therefore, *GT-D2G* can generate concept maps at scales not only without ground-truth training graphs but also without significant amounts of downstream task supervision.

5.5 Flexibility Evaluation (*RQ4*)

As discussed in §5.3, the $GT - D2G - var$ variant that is capable of generating flexible sizes of concept maps achieves the best document classification performance on two datasets (NYT and Yelp), while achieving the runner-up on the remaining dataset (AMiner). The observed experimental results justify the importance of the size-flexible property for concept map generation models.

To provide more insights, we further conduct experiments to explore the factors that impact the sizes of generated concept maps. As noted in the Training Techniques (§4.2), our framework is able to generate variable sizes of graphs by applying the RBF kernel-based graph size penalty and the content coverage penalty. These two penalties imply a trade-off between conciseness and completeness of generated concept maps. Fig. 7 shows the size distribution of the generated graphs on three datasets when the maximum graph size is set to be 10, 20, or 30 nodes. As can be seen, our *GT-D2G* can generate graphs with variable sizes as the size distribution varies according to the following two major factors: (a) input text complexity (across three datasets); (b) the preset hyperparameter "*max size*" (across different max sizes). For the input text complexity, we know that NYT and AMiner contain rather long and formal news articles and scientific reports, while Yelp contains short and informal online user-generated restaurant reviews. Consequently, concept maps derived from Yelp are inclined to have small sizes, while concept maps from NYT and AMiner



(a) NYT (b) AMiner (c) Yelp

Fig. 7: Graph size distributions on different max graph sizes.

have more evenly size distributions (when the *max size* is set to 30). For the hyperparameter *max size*, we can clearly see the set value bounds the actual sizes of generated graphs.

6 Conclusions

In this work, we aim to tackle the concept map generation task by graph translation networks. Without any gold training concept maps, the proposed *GT-D2G* framework is able to translate the initial concept maps into the target concise concept maps under the weak supervision from downstream tasks. The quality of generated concept maps is validated through both downstream task performance and human evaluation, in which *GT-D2G* outperforms other concept map generation methods by a wide margin. In the future, we plan to find more meaningful downstream tasks to demonstrate the effectiveness and generalizability of *GT-D2G*, and even study it in multi-task settings.

Acknowledgments

This is an extended and revised version of a preliminary conference short paper that was presented in SIGIR 2020 [12]. This work significantly expands the weakly supervised concept map generation model, and it provides additional analysis by manual evaluation, more compared methods, efficiency evaluation, and comprehensive ablation studies. This research is supported by the internal fund and GPU servers provided by the Computer Science Department of Emory University. Special thanks to Sophy Huang and Celia Hu for their generous help in the human annotation experiments.

References

[1] S.-H. Liu and G.-G. Lee, "Using a concept map knowledge management system to enhance the learning of biology," *Computers & Education*, 2013.

[2] X. Liu, L. Mian, Y. Dong, F. Zhang, J. Zhang, J. Tang, P. Zhang, J. Gong, and K. Wang, "Oag_know: Self-supervised learning for linking knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[3] T. Falke, C. M. Meyer, and I. Gurevych, "Concept-Map-Based Multi-Document Summarization using Concept Coreference Resolution and Global Importance Optimization," in *IJNLP*, 2017.

[4] T. Falke and I. Gurevych, "GraphDocExplore: A framework for the experimental comparison of graph-based document exploration techniques," in *EMNLP*, 2017.

[5] H. Cui, J. Lu, Y. Ge, and C. Yang, "How can graph neural networks help document retrieval: A case study on cord19 with concept map generation," in *ECIR*, 2022.

[6] J. D. Novak, "Concept mapping: A useful tool for science education," *Journal of research in science teaching*, 1990.

[7] S. L. Chen, T. Liang, M. L. Lee, and I. C. Liao, "Effects of concept map teaching on students' critical thinking and approach to learning and studying," *Journal of Nursing Education*, 2011.

[8] S. Bai and S. Chen, "Automatically constructing concept maps based on fuzzy rules for adapting learning systems," *Expert Syst. Appl.*, 2008.

[9] X. Huang, K. Yang, and V. B. Lawrence, "An efficient data mining approach to concept map generation for adaptive learning," in *ICDM*, 2015.

[10] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated Phrase Mining from Massive Text Corpora," *IEEE Trans. Knowl. Data Eng.*, 2018.

[11] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *EMNLP*, 2004.

[12] C. Yang, J. Zhang, H. Wang, B. Li, and J. Han, "Neural Concept Map Generation for Effective Document Classification with Interpretable Structured Summarization," in *SIGIR*, 2020.

[13] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *NeurIPS*, 2015.

[14] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *ICLR*, 2017.

[15] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *ICLR*, 2019.

[16] T. Falke and I. Gurevych, "Bringing Structure into Summaries: Crowdsourcing a Benchmark Corpus of Concept Maps," in *EMNLP*, 2017.

[17] V. Joshi, M. E. Peters, and M. Hopkins, "Extending a Parser to Distant Domains Using a Few Dozen Partially Annotated Examples," in *ACL*, 2018.

[18] H. He and J. D. Choi, "Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT," in *FLAIRS*, 2020.

[19] J. Lu and J. D. Choi, "Evaluation of unsupervised entity and event salience estimation," in *FLAIRS*, 2021.

[20] M. Simonovsky and N. Komodakis, "GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders," in *ICANN*, 2018.

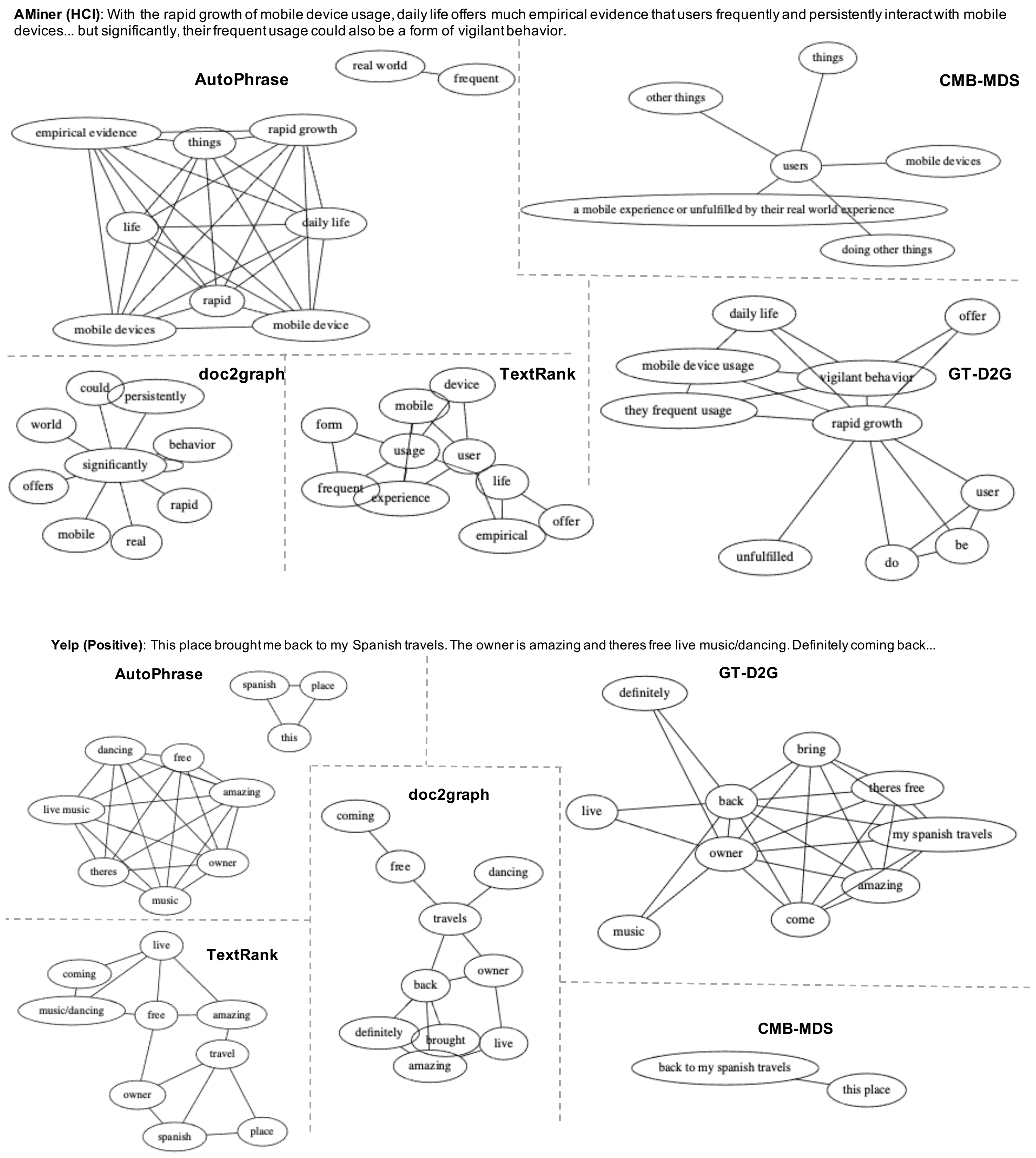
[21] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models," in *ICML*, 2018.

[22] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, "Graphaf: a flow-based autoregressive model for molecular graph generation," in *ICLR*, 2020.

[23] Y. Luo, K. Yan, and S. Ji, "Graphdf: A discrete flow model for molecular graph generation," in *ICML*, 2021.

[24] B. Chamberlain, J. Rowbottom, M. I. Gorinova, M. Bronstein, S. Webb, and E. Rossi, "Grand: Graph neural diffusion," in *ICML*, 2021.

[25] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling, "Equivariant diffusion for molecule generation in 3D," in *ICML*, 2022.



AMiner (HCI): With the rapid growth of mobile device usage, daily life offers much empirical evidence that users frequently and persistently interact with mobile devices... but significantly, their frequent usage could also be a form of vigilant behavior.

Yelp (Positive): This place brought me back to my Spanish travels. The owner is amazing and theres free live music/dancing. Definitely coming back...

Fig. 8: Concept maps generated by various models for case studies (*cont.*).

[26] C. Yang, L. Gan, Z. Wang, J. Shen, J. Xiao, and J. Han, "Query-Specific Knowledge Summarization with Entity Evolutionary Networks," in *CIKM*, 2019.
[27] W. Jin, K. Yang, R. Barzilay, and T. Jaakkola, "Learning multimodal graph-to-graph translation for molecule optimization," in *ICLR*, 2018.
[28] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[29] T. Zhao, X. Tang, X. Zhang, and S. Wang, "Semi-supervised graph-to-graph translation," in *CIKM*, 2020.
[30] W. Wang, Y. Gao, H.-Y. Huang, and Y. Zhou, "Concept pointer network for abstractive summarization," in *EMNLP-IJCNLP*, 2019.
[31] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *ACL*, 2014.
[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.
[33] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *EMNLP*, 2014.
[34] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling Coverage for Neural Machine Translation," in *ACL*, 2016.
[35] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *CoRR*, 2013.
[36] C. J. Maddison, A. Mnih, and Y. W. Teh, "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables," in *ICLR*, 2017.
[37] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization with Gumbel-Softmax," in *ICLR*, 2017.
[38] S. Chen, C. F. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on neural networks*, 1991.
[39] "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, 2005.
[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL-HLT*, 2019.
[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019.
[42] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

Jiaying Lu is a Ph.D. student in Computer Science at Emory University, advised by Prof. Carl Yang. He received his M.S. in Electronic and Communication Engineering and B.S. in Information Engineering at Beijing University of Posts and Telecommunications, China. His research interests include graph data mining, knowledge graph, and multimodal learning.

Xiangjue Dong is a Ph.D. student in Computer Science and Engineering at Texas A&M University, advised by Prof. James Caverlee. She earned her M.S. in Computer Science at Emory University in 2021 and M.S. in Civil Engineering at University of Illinois at Urbana-Champaign in 2019. Her research interests lie in natural language processing and conversational AI.

Carl Yang is an Assistant Professor of Computer Science in Emory University. He received his Ph.D. in Computer Science at University of Illinois, Urbana-Champaign in 2020, and B.Eng. in Computer Science and Engineering at Zhejiang University in 2014. His research interests span graph data mining, applied machine learning and structured information systems, with applications in knowledge graphs, recommender systems, biomedical informatics and healthcare. Carl's research results have led to over 50 publications in top peer-reviewed journals and conference proceedings including TKDE, KDD, WWW, NeurIPS, ICML, ICDE and SIGIR. He also received the Dissertation Completion Fellowship of UIUC in 2020 and the Best Paper Award of ICDM in 2020.