# Weakly Supervised Concept Map Generation through Task-Guided Graph Translation

Jiaying Lu
jiaying.lu@emory.edu
Emory University
Atlanta, Georgia

Xiangjue Dong
xiangjue.dong@emory.edu
Emory University
Atlanta, Georgia

Carl Yang
j.carlyang@emory.edu
Emory University
Atlanta, Georgia

arXiv:2110.15720v1 [cs.CL] 8 Oct 2021

## 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 document labels. The quality and interpretability of such concept maps are validated through human evaluation on three real-world corpora, and their utility in the downstream task is further demonstrated in the controlled experiments with scarce document labels.

## KEYWORDS

Concept Map Generation, Graph Translation, Weak Supervision

**ACM Reference Format:**
Jiaying Lu, Xiangjue Dong, and Carl Yang. 2021. Weakly Supervised Concept Map Generation through Task-Guided Graph Translation. In *Proceedings of (Under submission).* ACM, New York, NY, USA, 10 pages.

## 1 INTRODUCTION

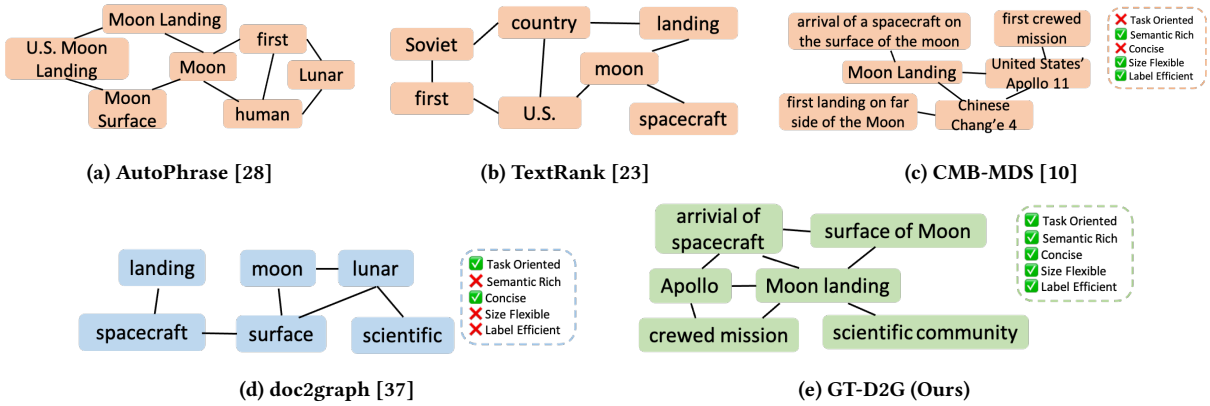
Standing out for the clear and concise structured knowledge representation, concept maps have been widely applied in knowledge management [19], document summarization [9, 10], and even as a research tool in educational science [5, 24]. Figure 1 shows toy examples of concept maps derived from a document 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 [1, 10, 13], 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 Figure 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 Figure 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* [37] 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 Figure 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 Section 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 traditional methods and *doc2graph*, 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 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*Under submission, ,*
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00




**(a) AutoPhrase [28]** **(b) TextRank [23]** **(c) CMB-MDS [10]**

**(d) doc2graph [37]** **(e) GT-D2G (Ours)**

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

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 Figure 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 Intial Graph Constructor to obtain a set of concept candidates with their associated relations. Then, a graph pointer network [31] based Graph Translator equipped with a graph convolution network [17] based Graph Encoder is applied upon the initial concept map to simultaneously select important concepts and links. A graph isomorphism network [35] 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 label 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

**Automatic Concept Map Generation**. The concept map generation task was first introduced by [8], where the task definition and a benchmark dataset *EDUC* was proposed. In [8], a corpus of 30 document clusters in which each contains around 40 source document 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*, the extended model from same research group, [10] 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* [37] did not rely on human-generated concept maps as training samples. Instead, the graph generation model in doc2graph was trained with weak supervisions from downstream tasks.

**Graphical Structures for Texts**. Graph structures have been extensively used in various parsing tasks, including keyword based graph [23, 28] and parsing-based graph [2, 12, 16]. *TextRank* [23] 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* [28] 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-occuring 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 to provide the semantic-rich initial graph and propose a graph translation model to enable the size flexible concept map generation.

**Neural Networks for Graph Generation**. Inspired by the success of CNNs [18] in computer vision field, various graph neural networks [17, 27, 32] for encoding graphical structures into dense, real-valued vector embeddings have been developed in parallel. By

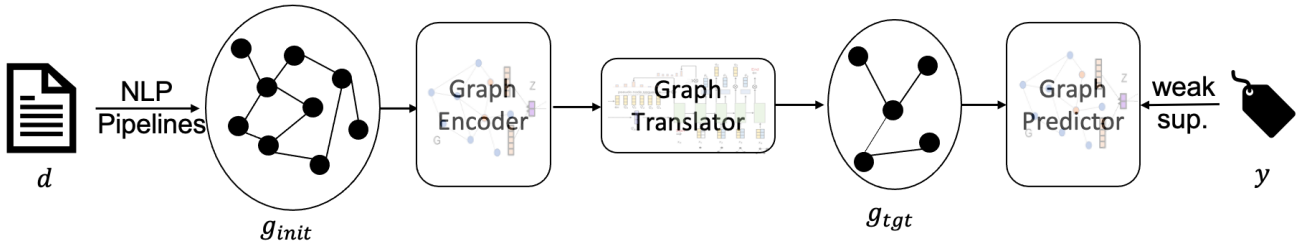



**Figure 2: Overview of proposed *GT-D2G* framework.**

recursively aggregating and transforming representation vectors of neighbour vertices, GNNs have achieved significant progress on node classification, link prediction, graph classification, community detection, and many other graph-related tasks. The focus of this paper, concept map generation, is a sub-task of the graph generation task. While GraphVAE [29], GraphRNN [38], CondGen [36] etc. have shown a great ability of graph generation task in which the goal is to reconstruct graphs from the high-dimensional vector, the goal of concept map generation task is to construct a local knowledge graph from raw text. Therefore, some variants of GNNs could ideally model for concept map generation task.

## 3 PROBLEMS & MOTIVATIONS

**Problem Definition**. Our goal in this work is to predict the labels $\hat{\mathcal{Y}}$ of text-related downstream tasks for an input corpus $\mathcal{D}$ with gold labels $\mathcal{Y}$, while simultaneously generating concept maps $\mathcal{G}$ to provide interpretability towards the structural summarization of $\mathcal{D}$ regarding the main concepts and their interactions. The downstream tasks can be flexible, possibly ranging from document-level classification, retrieval to sentence-level natural language inference and relation extraction. We formulate the generated concept map as a unified local graph that can distill and represent the key knowledge for each input text $d_i$, where $d_i \in \mathcal{D}$ is a sequence of words, i.e., $d_i = (w_1^i, w_2^i, \ldots, w_{l_i}^i)$. The concept maps $\mathcal{G}$ are defined as local graphs, hence for each $d_i$ we aim to generate an associated graph $g_i = \{C^i, \mathcal{M}^i\}$ which focuses on the concepts $C^i$ and their interactions $\mathcal{M}^i$ in the span of $d_i$. $C$ is a set of $n$ concepts that can be words, phrases or sentences depending on the downstream tasks, and $\mathcal{M} \subset \mathcal{R}^{n \times n}$ indicates the interaction strength (i.e. edge weight) among concepts in $C$.

The evaluation metrics include both metrics $f(\mathcal{Y}, \hat{\mathcal{Y}})$ for downstream NLP tasks and the quality $h(\mathcal{G})$ of generated graphs. Due to the sparsity of gold concept maps, $h(\mathcal{G})$ typically involves a human evaluation process. Moreover, the downstream task performance partially conveys the quality of generated concept maps. In this project, we focus on document classification as the particular downstream task for quantitative evaluation, and we leave the study of other possible tasks as future work.

Recent advances in concept map generation methods, such as *CMB-MDS* [10] and *doc2graph* [37], have made important progress towards automatically generating large scale of concept maps from text data. However, these recently proposed models are either deviating from the focus of downstream tasks or limited to generating semantically incomplete concept maps, due to the following challenges.

- **How to build semantic rich graphs?** Semantic rich graphs provide valuable interpretability for humans to better understand the decisions made by neural networks, as well as quickly grasp the structured knowledge inside natural texts. Although achieving higher accuracy than existing models, doc2graph has been criticized for its shortage in producing nodes with poor semantic meanings due to the limitation of directly picking words from raw texts as concepts. In the meantime, it fails to leverage many existing available NLP tools to build graphical structures for texts, which have shown great benefits to provide rich information and boost performance on text-related tasks [15, 22]. To this end, we leverage the generalized NLP pipelines to build semantic rich graphs as higher starting points for *GT-D2G*.
- **How to construct task oriented graphs?** While many traditional methods also utilize NLP pipelines, these methods can hardly construct task oriented graphs. Inspired by the semi-supervised training diagram proposed by doc2graph, *GT-D2G* interpret the problem of concept map generation as translating the semantic rich initial graphs through a task guided data-driven learning process, thus addressing the task deviating issue. In addition, the utilization of task supervision enables *GT-D2G* to be applied to sample sparse datasets or domains.

To sum up, we propose *GT-D2G* as a weakly supervised concept map generation framework that leverages an unsupervised NLP pipeline to build semantic rich graphs, after which the initial graphs are then translated to target graphs guided by limited task signal. The translated graphs can be regarded as concisely structured summarization of the input text with valuable interpretability, and can be directly used by downstream tasks to improve the prediction performance. Specifically, we do not aim to surpass state-of-the-art document classification neural models such as BERT [7] that take very large corpora as input and tremendous computational resources. Instead, the focus of *GT-D2G* is to generate concept maps for input documents that provide concise, interpretable structured summary while outperforming other existing graph-based baselines.

## 4 PROPOSED APPROACH

Figure 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 $Q_i$, 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 [37] is that single words are directly picked from the raw texts through a Pointer Network [31] and considered as nodes in the final concept map. However, words purely picked by a simple Pointer Network can easily be of low quality [34]. Moreover, phrases are often preferable to represent concepts, especially noun phrases as semantically complete concepts [28]. 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 leverage existing NLP pipelines. 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 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, coreference resolution technique is used to resolve all pronoun expressions in documents. We use the popular Stanford CoreNLP [21] for all steps mentioned above.

**Link Generation.** For links between extracted nodes, we follow the sliding window idea introduced in keyphrase extraction studies [23]. 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 other algorithms can be applied to construct weighted links or directed links, e.g., directly using the parsing tree to construct the graph or only constructing certain types of links through relation extraction. The graph ensemble process is trivial once both nodes and links are extracted.

## 4.2 Task Guided Graph Translation

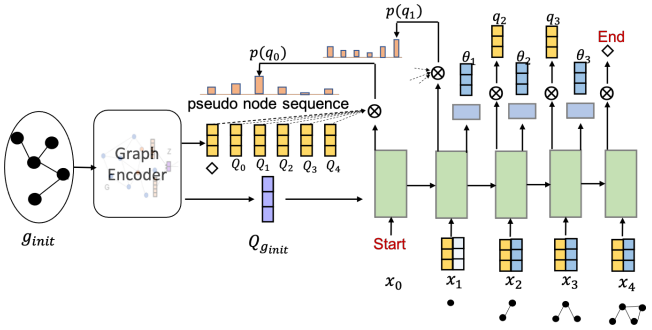



**Figure 3: Proposed Graph Translator. Green rectangles denote RNN cells that take the previous time step chosen node $q_{t-1}$ and generated adjacency vector $\theta_{t-1}$ as input. The RNN state vector $h_t$ is updated at every time step, and is initialized by the graph level representation of initial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

**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) [17] as our Graph Encoder. The node embeddings $Q^{(k+1)}$ are learned after the $k$-th layer of GCN by the following equation

$$Q^{(k+1)} = ReLU(\tilde{D}^{-\frac{1}{2}}\tilde{M}\tilde{D}^{-\frac{1}{2}}Q^{(k)}W_q^{(k)}), \quad (1)$$

where $W_q^{(k)}$ is learnable parameters in the $k$-th layer of GCN, $\tilde{M}$ is the adjacency matrix $M_{init}$ with additional self-connections and $\tilde{D}_{ii} = \sum_j \tilde{M}_{ij}$ is the diagonal degree matrix. The input node embeddings $Q^{(0)}$ are the concatenations of phrase embeddings, normalized frequency feature and normalized location feature. The phrase embedding of each node is the average of pretrained word embeddings (in practice, we use GloVe [26]). The frequency feature and the location feature reflect the importance of the 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 $Q_{g_{init}} = \frac{1}{n}\sum_{i=1}^{n} Q_i$ to encode the global contextual information in the whole 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 $Q_i$ ($i \in [1, n]$) and graph-level embedding $Q_{g_{init}}$ 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 [31] 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 [38] 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 [31] 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 Figure 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 Figure 3. In practice, we choose GRU [6] as the implementation. In order to start the translation from the whole initial graph, the hidden state of the RNN decoder is initialized by $h_0 = Q_{g_{init}}$. Therefore, the hidden state that encodes the "graph translation state" is updated by

$$h_t = f_{GRU}(x_t, h_{t-1}), \tag{2}$$

where $h_{t-1}$ denotes the hidden state from the last time step, and $x_t$ denotes the input at the current time step. More specifically, we compute $x_t$ as the representations of both nodes and links generated from the last time step, which can be denoted as $x_t = [q_{t-1}, \theta_{t-1}]$, where $[\cdot, \cdot]$ denotes vector concatenation. $q_{t_1}$ is the node embedding $Q_i$ from the Graph Encoder of the last selected node $i$, and we defer the explanation towards $\theta_{t-1}$ to the later part of this subsection. It is worth mentioning that $x_0$ is a zero vector that represents the starting point of sequence generation.

*Deeply coupled node and link generation.* Once we obtain the RNN decoder hidden state $h_t$, the node selection process can be described by the following equations

$$u_i^t = v^T tanh(W_1 Q_i + W_2 h_t), \tag{3}$$

$$p(q_t = Q_i | q_1, ..., q_{t-1}, Q_{g_{init}}) = softmax(u_i^t), \tag{4}$$

where softmax normalizes the real-valued vector $u^t$ (of length equal to the input graph size $n$) to be an attention vector of probability distribution over all nodes in the input graph, and $v, W_1, W_2$ are learnable parameters. Our GPT then selects the $i$-th node from the initial graph with maximum pointer attention $i = \text{argmax}_i(p(q_t = Q_i))$, adding the node into the translated target graph and feed $q_t = Q_i$ 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 [30], by maintaining a coverage vector $c_t = \sum_{t'=0}^{t-1} p(q_{t'})$ that accumulates the generated attentions so far, while adding the following loss to enforce the model to pay more attention to nodes not covered yet:

$$L_{cov} = \sum_{d_i \in \mathcal{D}} \sum_{t_j \in d_i} min(p(q_{t_j}), c_{t_j}). \tag{5}$$

To deeply couple the generation process of nodes and links so that the target graph (i.e., final concept map) is meaningful as a whole, we get inspired by the recent deep graph generation model of GraphRNN [38]. 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 Figure 3 and described in the following equation

$$\theta_t = f_{out}(h_t), \tag{6}$$

where $\theta_t$ 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 RNN decoder. On the other hand, the *neigh* variant interprets the generation problem as generating all possible neighbours 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 numbers 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 and adjacency vectors, the assembling of target graph is trivial. Furthermore, a *Graph Isomorphism Network* (GIN) [35] is adopted as our Graph Predictor which is directly applied to predict the document-level class labels. Specifically, the GIN we implement has the same architecture as the GCN described in Equation 1, but with an additional two layer MLP attached to predict the document category $\hat{y}$ after using the pooling read-out function for graph-level representation:

$$\hat{y} = MLP(pooling(Q'_1, Q'_2, \ldots)), \tag{7}$$

where $\{Q'_k\}$ are the node-level embeddings in GIN for the final concept nodes in the target graph $g_{tgt}$ produced by our Graph Translator.

**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

$$L_{cls} = -\sum_{d_i \in \mathcal{D}} p(\hat{y_i}) \log p(y_i), \tag{8}$$

$$L = L_{cls} + \lambda * L_{cov}, \tag{9}$$

where $\lambda$ is a tunable hyper-parameter.

One technical challenge exists for the node selection operation that selects the node with maximum pointer attention $i =$

$\text{argmax}_i(p(q_t = Q_i))$ during the graph translation process in GPT. As we know, the max value selection operation implemented as *argmax* [*] is non-differentiable, thus leading to the lost gradient after node selection. In a classic seq-to-seq setting, the selection operation only impacts the subsequent sequence generation, and the supervision signals can be propagated since the ground-truth sequence is given. However, in our *GT-D2G* setting, we need to propagate gradients all the way from the Graph Predictor in the end to the Graph Translator and Graph Encoder in the beginning. Inspired by the Straight-Through Gumbel Softmax trick [3, 14, 20], we use a hard-version *Gumbel-Softmax* [†] to sample one-hot vectors from the predicted probabilities, so that the *argmax* process is fully differentiable.

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 Figure 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

$$L_{len} = \sum_{d_i \in \mathcal{D}} \sum_{t_j \in d_i} Penalty(t_j) \cdot P(q_{t_j} = \text{"}EOS\text{"}). \tag{10}$$

The function $Penalty(t) > 0$ 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 [4]. Therefore, the overall loss function for *GT-D2G* is:

$$L = L_{cls} + \lambda_1 * L_{cov} + \lambda_2 * L_{len}. \tag{11}$$

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.

## 5 EXPERIMENTS

In this section, we evaluate our proposed *GT-D2G* framework focusing on the following four research questions:

(1) ***RQ1***: How is the quality of *GT-D2G* generated graphs?
(2) ***RQ2***: How do *GT-D2G* and its variants perform in comparison to other document classification methods?
(3) ***RQ3***: Is *GT-D2G* label efficient?
(4) ***RQ4***: Can *GT-D2G* generate graphs with flexible sizes depending on the complexity of input text?

### 5.1 Experiment Settings

**Datasets**. Our experiments are conducted on three real-world text corpora: *NYTimes*, *AMiner*, and *Yelp*. Different from *doc2graph* [37], 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 [37] to randomly split the labeled documents into 80% for training, 10% for validation, and 10% for testing.

| Dataset | #doc | #word | #category | Init Concept Map | | |
|---|---|---|---|---|---|---|
| | | | | #node | #edge | #degree |
| NYTimes | 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 |

**Table 1: Statistics of three datasets.**

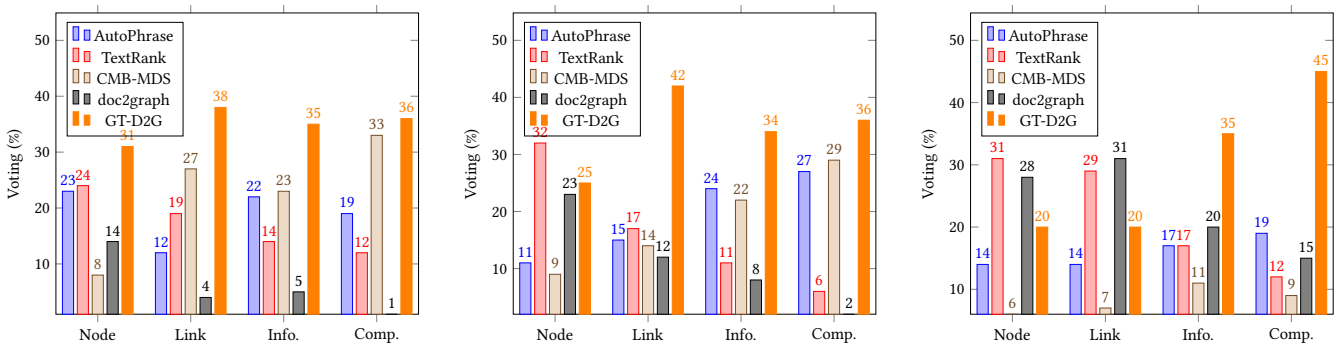
We choose accuracy as the metric for document classification tasks. To get a stable result, we run each model three times and report the mean ± standard deviation.

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

- **Graph-Based Methods** as major competitors.
  - ***AutoPhrase*** [28]: 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*** [23]: 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*** [10]: 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*** [37]: *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*** [11]:*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.
  - ***BERT-base*** [7]: *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 [25] and DGL [33], with code publicly available. Implementations of the compared baselines are either from open-source project (BERT[‡]) or the original authors (Bi-LSTM/ AutoPharse/ TextRank/ CMB-MDS/ doc2graph[§]). We optimize *GT-D2G* through the Adam optimizer with learning rate to $3e-4$ and max epoch to 500. The temperature parameter $\tau$ 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

[*] https://pytorch.org/docs/stable/generated/torch.argmax.html
[†] https://pytorch.org/docs/stable/_modules/torch/nn/functional.html#gumbel_softmax
[‡] https://github.com/huggingface/transformers
[§] https://github.com/JieyuZ2/doc2graph




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

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

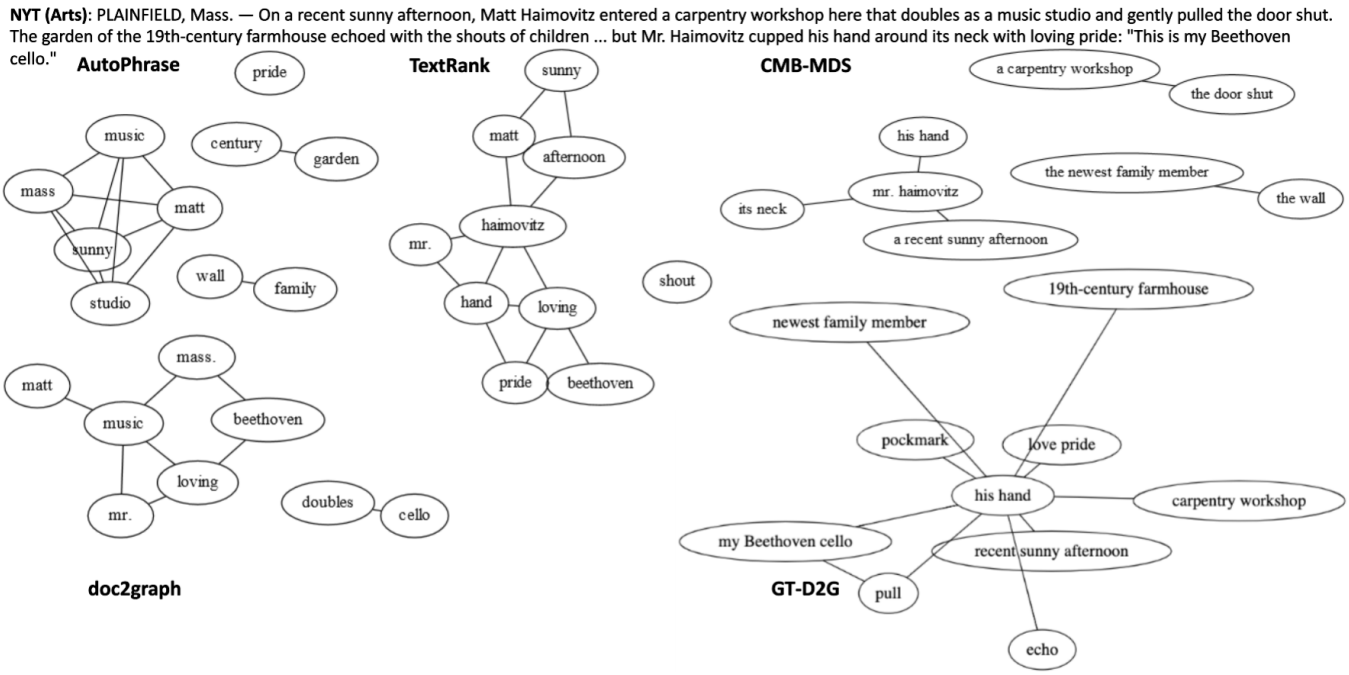



**Figure 5: Case study on NYT. Due to the page limitation, more case studies can be viewed at the link (link is masked during double-blind review).**

GCN, RNN and MLP are set to 128, and the number of GNN layers in all GCN, GIN models are 2 with average as the pooling function. For RBF kernel function used to penalize overlength node sequence, $\sigma$ 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*)

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

| 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 |

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

to pick k (k<= 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 Figure 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 Figure 5. 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., *"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*)

| Algorithm | NYT | AMiner | Yelp |
|---|---|---|---|
| **Bi-LSTM** | 87.52 ± 3.01 | 59.32 ± 2.71 | 78.46 ± 1.46 |
| **BERT-base** | 97.54 ± 0.16 | 73.62 ± 0.06 | 85.34 ± 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 | **68.53** ± 1.02 | 80.92 ± 0.50 |
| **GT-D2G-var** | **95.46** ± 0.49 | 68.37 ± 1.05 | **80.98** ± 0.51 |

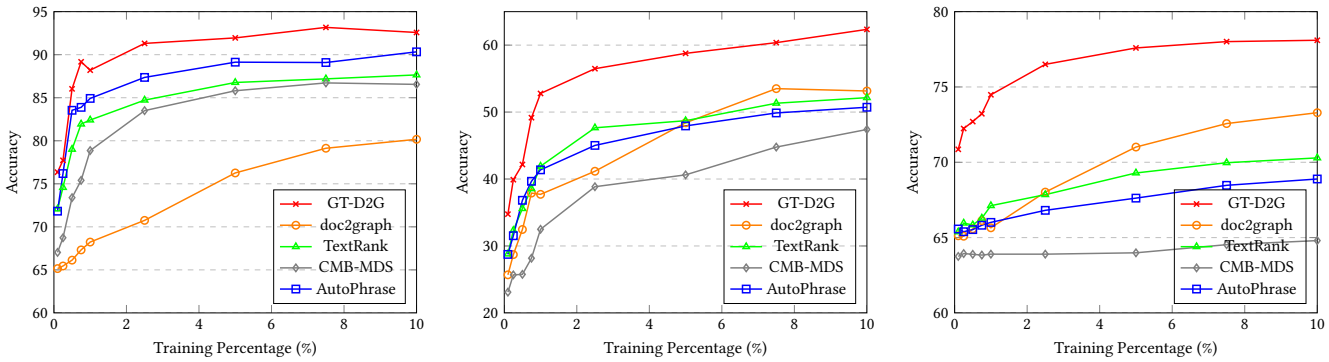
**Table 3: Document classification accuracy(%).**

To answer *RQ2*, we conduct the document classification experiments on three text corpora (Section 5.1). 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 through weak supervisions. 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. To better understand the effectiveness of our proposed techniques (Section 4) , we closely study the four ablations of *GT-D2G* regrading effectiveness of NLP pipelines (*-init*), node-and-link iterative generation (*-path* and *-neigh*), and flexible-size graph generation (*-var*).

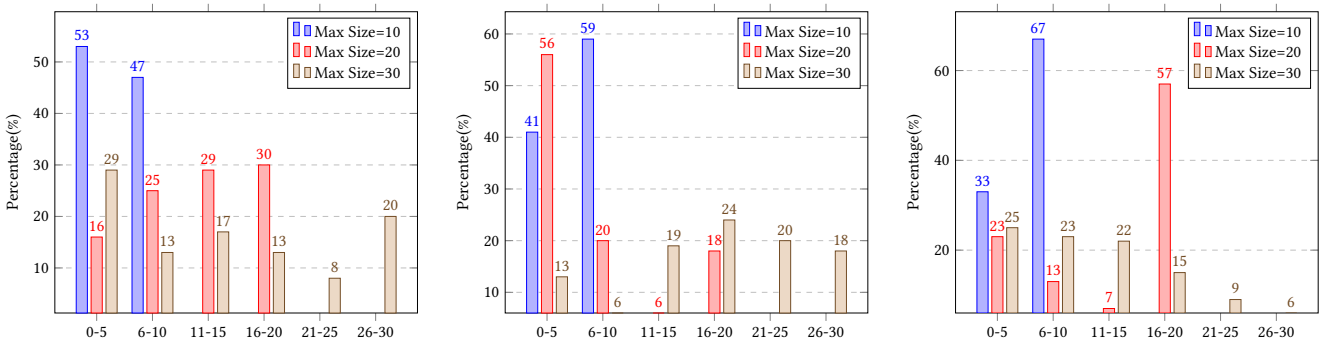
In particular, to evaluate the effectiveness of incorporating NLP pipelines, we implement *GT-D2G-init* that directly encodes all nodes in the initial semantic-rich concept maps to make predictions. The experiment results in Table 3 show that *GT-D2G-init* outperforms all traditional baselines on three datasets and outperforms *doc2graph* on *NYT* and *Yelp*, which demonstrate that our initial concept maps are helpful for the downstream tasks. The significant improvement on *NYT* may be due to the entity&event-centric characteristic of news documents, which is more effectively captured by our initial graph; while the initial graphs are less effective on scientific papers and custom reviews. Upon *GT-D2G-init*, other three ablations add Graph Translator module to obtain a more concise concept map, since the initial concept map often contains 20-40 nodes and the translated concept map contains less than 10 nodes. In addition, the translated concept map is more effective as it achieves higher document classification accuracy.

To explore 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 neighbours. From the results shown in Table 3, *GT-D2G-neigh* is consistently better on all three datasets, which well supports our argument that generating edges among all possible neighbours is preferable than generating edges as a sequence of path 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 details about the flexibility are answered in RQ4 (Section 5.5).




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

**Figure 6: Test accuracy by varying the proportions of training data (0.1%, 0.25%, 0.50%, 0.75%, 1.00%, 2.50%, 5.00%, 7.50%, 10.00%).**




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

**Figure 7: Graph size distribution on different max graph sizes.**

## 5.4 Label Efficiency Evaluation (*RQ3*)

To demonstrate the label 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 trails of each experiment by applying different random seeds. The average test accuracies of *NYTimes*, *AMiner*, and *Yelp* datasets were shown in Figure 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 the excellent performance during the whole label efficiency evaluation with limited labeled data. These results demonstrate the label efficiency of our model. Therefore, *GT-D2G* is able to generate concept maps at scale not only without ground-truth training graphs, but also without significant amounts of downstream task supervisions.

## 5.5 Flexibility of Generating Different Size of Graphs (*RQ4*)

To provide more insights on the flexibility of our proposed *GT-D2G*, we further conduct experiments aiming at generating concept maps with variable sizes (*GT-D2G-var*). As noted in the Training Techniques (Section 4.2), our framework is able to generate variable size 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.

Figure 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. Furthermore, we can see that the size of generated graphs distributes pretty evenly when different maximum graph sizes are set for the experiments on *NYT* and *AMiner*; while on *Yelp*, the generated graphs are prone to be smaller, possibly due to the short and informal characteristics of reviewer-generated text data.

## 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.

# REFERENCES


[1] Shih-Ming Bai and Shyi-Ming Chen. 2008. Automatically constructing concept maps based on fuzzy rules for adapting learning systems. *Expert Syst. Appl.* 35, 1-2 (2008), 41–49.

[2] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* (2013).

[4] Sheng Chen, Colin FN Cowan, and Peter M Grant. 1991. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks* 2, 2 (1991), 302–309.

[5] Shiah Lian Chen, Tienli Liang, Mei Li Lee, and I Chen Liao. 2011. Effects of concept map teaching on students' critical thinking and approach to learning and studying. *Journal of Nursing Education* 50, 8 (2011), 466–469.

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

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

[8] Tobias Falke and Iryna Gurevych. 2017. Bringing Structure into Summaries: Crowdsourcing a Benchmark Corpus of Concept Maps. In *EMNLP*.

[9] Tobias Falke and Iryna Gurevych. 2017. GraphDocExplore: A Framework for the Experimental Comparison of Graph-based Document Exploration Techniques. In *EMNLP*.

[10] Tobias Falke, Christian M. Meyer, and Iryna Gurevych. 2017. Concept-Map-Based Multi-Document Summarization using Concept Coreference Resolution and Global Importance Optimization. In *IJNLP*.

[11] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18, 5 (2005), 602–610. IJCNN 2005.

[12] Han He and Jinho D. Choi. 2020. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *FLAIRS*.

[13] Xiaopeng Huang, Kyeong Yang, and Victor B Lawrence. 2015. An efficient data mining approach to concept map generation for adaptive learning. In *ICDM*.

[14] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.

[15] Ming Jiang and Jana Diesner. 2019. A Constituency Parsing Tree based Method for Relation Extraction from Abstracts of Scholarly Publications. In *TextGraphs-13*.

[16] Vidur Joshi, Matthew E. Peters, and Mark Hopkins. 2018. Extending a Parser to Distant Domains Using a Few Dozen Partially Annotated Examples. In *ACL*.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*.

[19] Shih-Hwa Liu and Gwo-Guang Lee. 2013. Using a concept map knowledge management system to enhance the learning of biology. *Computers & Education* 68 (2013), 105–116.

[20] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.

[21] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.

[22] David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event Extraction as Dependency Parsing. In *ACL*.

[23] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*.

[24] Joseph D Novak. 1990. Concept mapping: A useful tool for science education. *Journal of research in science teaching* 27, 10 (1990), 937–949.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*.

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

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*.

[28] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. 2018. Automated Phrase Mining from Massive Text Corpora. *IEEE Trans. Knowl. Data Eng.* (2018).

[29] Martin Simonovsky and Nikos Komodakis. 2018. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. In *ICANN*.

[30] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *ACL*.

[31] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NeurIPS*.

[32] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *SIGKDD*.

[33] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[34] Wenbo Wang, Yang Gao, He-Yan Huang, and Yuxiang Zhou. 2019. Concept pointer network for abstractive summarization. In *EMNLP-IJCNLP*.

[35] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.

[36] Carl Yang, Lingrui Gan, Zongyi Wang, Jiaming Shen, Jinfeng Xiao, and Jiawei Han. 2019. Query-Specific Knowledge Summarization with Entity Evolutionary Networks. In *CIKM*.

[37] Carl Yang, Jieyu Zhang, Haonan Wang, Bangzheng Li, and Jiawei Han. 2020. Neural Concept Map Generation for Effective Document Classification with Interpretable Structured Summarization. In *SIGIR*.

[38] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*.