Neural Concept Map Generation for Effective Document Classification with Interpretable Structured Summarization

Carl Yang*[1], Jieyu Zhang*[2], Haonan Wang[2], Bangzheng Li[2], Jiawei Han[2]
[1]Emory University,[2]University of Illinois at Urbana Champaign
[1]j.carlyang@emory.edu, [2]{jieyuz2, haonan3, bl17, hanj}@illinois.edu

ABSTRACT

Concept maps provide concise structured representations for documents regarding their important concepts and interaction links, which have been widely used for document summarization and downstream tasks. However, the construction of concept maps often relies heavily on heuristic design and auxiliary tools. Recent popular neural network models, on the other hand, are shown effective in tasks across various domains, but are short in interpretability and prone to overfitting. In this work, we bridge the gap between concept map construction and neural network models, by designing *doc2graph*, a novel weakly-supervised text-to-graph neural network, which generates concept maps in the middle and is trained towards document-level tasks like document classification. In our experiments, *doc2graph* outperforms both its traditional baselines and neural counterparts by significant margins in document classification, while producing high-quality interpretable concept maps as document structured summarization.

ACM Reference Format:
Carl Yang, Jieyu Zhang, Haonan Wang, Bangzheng Li, Jiawei Han. 2020. Neural Concept Map Generation for Effective Document Classification with Interpretable Structured Summarization. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401312

1 INTRODUCTION

Structured knowledge has been studied for information extraction and retrieval for a long time [2, 15]. Among various types of structures, concept maps stand out as natural interpretable representations of texts (*i.e.*, single documents [18], multi-documents [3, 4]), and have been shown effective for document summarization [7, 8] and various text-related downstream tasks [6, 11]. A concept map as we discuss in this work is a graphical representation of important knowledge distilled from the source document, where each node in the graph is a concept, and each link indicates the interaction between two concepts. Figure 1 gives two toy examples of short documents and their corresponding concept maps.

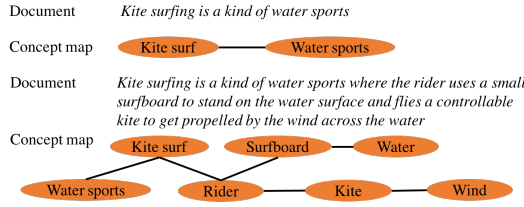

Figure 1: Toy examples of documents and concept maps.

Existing research. To construct concept maps from documents, existing works mostly follow the multiple steps of concept extraction, relation extraction, scoring and assembling. They assume various a-priori knowledge and leverage ad hoc approaches for each step, such as auxiliary data like rules, dictionaries, ontologies, and additional tools like lexical parsing, POS tagging, term disambiguation, which are not always available and effective, and can hardly generalize across tasks, domains and languages [1, 4, 18]. Moreover, the multiple steps are often conducted separately, blocking possible mutual enhancement and can hardly lead to optimal performance towards any particular downstream task.

On the other hand, recent extensive research on neural network (NN) models has demonstrated superior performance in various domains. In particular, neural generative models like VAEs and GANs can generate high-quality images, texts and graphs [16, 17]. However, no previous work has considered the direct generation of graphs from texts with NN models. Moreover, NN models have been constantly criticized for their shortage in interpretability and vulnerability towards overfitting, due to the common brute increase of model complexity and lack of proper regularization.

Present work. For the first time, we propose to learn to generate concept maps within an end-to-end NN model. In particular, we design a novel autoencoder framework to generate concept maps (*i.e.*, concept nodes and their interaction links) from text data (*i.e.*, documents), and train it with weak supervision from text-related downstream tasks (*e.g.*, document classification). In the meantime, the generated concept maps as intermediate results serve as proper regularization against overfitting to improve the downstream task and can be readily used as structured summarization of the source documents with valuable interpretability.

Through experiments on three real-world text corpora with a primitive configuration of the neural architectures, we demonstrate that our proposed *doc2graph* framework can effectively improve document classification regarding its non-neural traditional concept map generation baselines, as well as its neural non-graph-generation counterparts. Since it is hard to quantitatively evaluate the quality of concept maps [3, 18], we also conduct multiple real case studies to showcase that *doc2graph* generates plausible concept maps as interpretable document structured summarizations.

*Both authors contribute equally.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401312

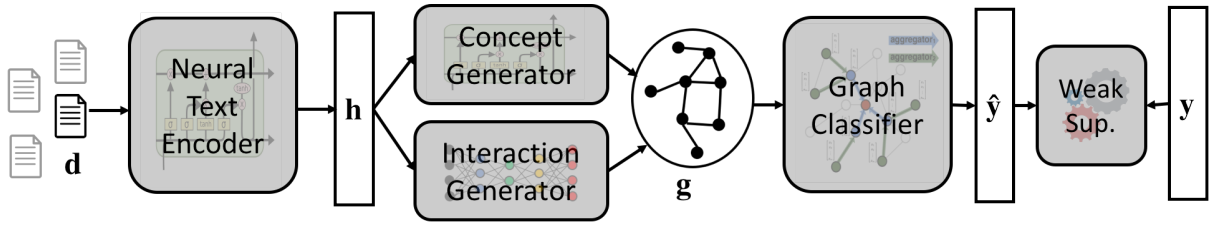

Figure 2: Overview of our proposed *doc2graph* neural framework: A neural text model encodes the content of each document $\mathbf{d}$ into an embedding vector $\mathbf{h}$. A concept generator and an interaction generator use $\mathbf{h}$ to generate concepts C and weighted links $\mathbf{M}$ in parallel, which are directly combined as a concept map $\mathbf{g}$. A graph classifier predicts a document label $\hat{\mathbf{y}}$ based on $\mathbf{g}$, which can be directly trained towards the ground-truth document label $\mathbf{y}$, while providing weak supervision to the intermediate generation of $\mathbf{g}$.

2 *DOC2GRAPH*

2.1 Problem Definition

Our input is a text corpus $\mathcal{D}$ with some labels $\mathcal{Y}$ of certain downstream text-related tasks, such as document classification, ranking, pairwise link prediction, *etc.* For simplicity, we focus on document classification in this work. Available document labels $\mathcal{Y}$ partitions $\mathcal{D}$ into a set of labeled documents $\mathcal{D}_l$ and unlabeled documents $\mathcal{D}_u$. Each document $d_i \in \mathcal{D}$ is a sequence of words, *i.e.*, $d_i = (w_1^i, w_2^i, \cdots, w_{l_i}^i)$, where l_i is the length of d_i.

Our first output is the predicted class labels $\hat{\mathcal{Y}}$ for unlabeled documents $\mathcal{D}_u$, which conforms to the standard task of supervised document classification. Our second output is the generated concept maps $\mathcal{G}$ for all documents $\mathcal{D}$ in the corpus, which is our novel contribution. Ideally, $\mathcal{G}$ should concisely distill and represent the key knowledge in $\mathcal{D}$, so as to serve as document structured summarization. In particular, we focus on the main concepts and their interactions in the single-document level. That is, we aim to generate a graph $g_i = \{C^i, \mathbf{M}^i\} \in \mathcal{G}$ for each document $d_i \in \mathcal{D}$, where C^i is a set of n_i concepts (*e.g.*, words, phrases) and $\mathbf{M}^i \subset \mathbb{R}_0^{+n_i \times n_i}$ denotes the interaction strength among concepts in C^i.

For simplicity, we only care about the concept interaction strength in this work, while our proposed neural framework is flexible enough to also generate and associate relation words or phrases to the concept links in the future.

2.2 Model Details

Figure 2 and its caption give an overview of our proposed framework. In this work, we do not focus on competing with the state-of-the-art complicated NLP models either for document classification or summarization. Instead, we aim to highlight the novel design of our proposed overall neural framework and show a possibility of (1) improving neural document classification by using concept maps as intermediate regularization; (2) enabling automatic document structured summarization under the weak supervision of classification. For simplicity concern, we intentionally design the most straightforward yet effective neural architectures for each module in our framework, which can be further improved in the future.

Neural text encoder. We implement the text encoder with *bidirectional Long Short-Term Memory networks* (bi-LSTM), which has been widely used to encode sequential data including texts, due to its known ability of capturing long-term dependencies between tokens in source text. It converts each variable-length input sequence $d = (w_1, w_2, \cdots, w_l)$ into a sequence of encoded hidden states $\mathbf{h}_t$ through the modeling of sequentially organized input w_t, output $\mathbf{o}_t$, forget gate $\mathbf{f}_t$ and memory cell $\mathbf{c}_t$. For the connection to subsequent modules as well as the training and evaluation of this base model, we compute the final text embedding vector of d as $\mathbf{h} = (1/l)\sum_{t=1}^{l} \mathbf{h}_t$. We follow the standard design of bi-LSTM and omit the detailed structure here.

Concept generator. Our concept generator aims to select a set of n concepts C in terms of keywords or keyphrases based on each document d (we fix n as a small value like 10 across all documents, while it can be also made flexible in the future). Since this task is essentially similar to keyword based summarization, we simplify the widely used *pointer networks* [13] to our setting of no predefined dictionaries, by only allowing its *copying mechanism* to select informative tokens from the source document. In particular, a *one-directional LSTM* takes $\mathbf{h}$ as the first input and sequentially generate n tokens as concepts in C based on the following attention mechanism

$$\mathbf{e}_i^t = \mathbf{w}^T \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_s \mathbf{s}_t + \mathbf{b}),\ \mathbf{a}^t = \text{softmax}(\mathbf{e}^t), \qquad (1)$$

where $\mathbf{w}, \mathbf{W}_h, \mathbf{W}_s, \mathbf{b}$ are learnable parameters. $\mathbf{h}_i$'s are the hidden states corresponding to tokens in the source document computed by the bi-LSTM encoder, $\mathbf{s}_t$'s are the hidden states of the LSTM concept generator corresponding to each generated token c_t. Each attention distribution $\mathbf{a}^t$ is used to select a token from the source document as the next generated token c_{t+1}. For training and evaluation purposes, we compute the *context vector* $\mathbf{h}_t^* = \sum_{i=1}^{l} \mathbf{a}_i^t \mathbf{h}_i$ of each generated token and compute the overall embedding of generated tokens (concepts) as $\mathbf{h}^* = (1/n)\sum_{t=1}^{n} \mathbf{h}_t^*$. To alleviate the common repetition problem for sequence-to-sequence models, we also adapt the *coverage model* proposed in [12], by maintaining a *coverage vector* $\mathbf{c}^*$ to both inform the concept generator of its previous decisions and penalize repetitive selections.

Interaction generator. Our interaction generator aims to construct a weighted none-negative matrix $\mathbf{M} \subset \mathbb{R}_0^{+(n \times n)}$ that captures the interaction strengths among the n concepts in C. To this end, we get inspired by recent works on neural graph generation, and in particular, we choose the straightforward design of a simple 2-layer *multi-layer perceptron* (MLP) to directly compute $\mathbf{M}$ from the output of text encoder $\mathbf{h}$. In particular, we aim to generate undirected links among the n concepts and thus compute only $n(n-1)/2$ elements to construct a symmetric matrix $\mathbf{M}$ without self-loops

$$\mathbf{m} = \text{sigmoid}\big(\mathbf{W}_m^{(2)} \text{ReLU}(\mathbf{W}_m^{(1)} \mathbf{h} + \mathbf{b}_m^{(1)}) + \mathbf{b}_m^{(2)}\big), \qquad (2)$$

where $\mathbf{W}_m, \mathbf{b}_m$ are learnable parameters in the two layers of MLP, and the last layer is of size $n(n-1)/2$.

Graph classifier. After generating the n concepts C and $n \times n$ interaction matrix $\mathbf{M}$, we directly combine them into a concept map g. For training and evaluation purposes, we adopt the popular *graph convolutional network* (GCN) [5] with the *mean read-out function* for graph-level classification [14], which is briefly described as follows

$$\mathbf{Q}^{(k+1)} = \text{ReLU}\Big(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{M}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Q}^{(k)}\mathbf{W}_q^{(k)}\Big), \tag{3}$$

where $\mathbf{W}_q^{(k)}$ is the learnable parameter in the kth layer of GCN. $\tilde{\mathbf{M}}$ is the generated link matrix $\mathbf{M}$ with additional self-connections, and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{M}}_{ij}$. We use the context vectors of generated concepts as the input node features of GCN, *i.e.*, $\mathbf{Q}^{(0)} = \mathbf{H}^*$. We compute a graph-level embedding by averaging all node-level embedding output by GCN as $\mathbf{q}^* = (1/n)\sum_{t=1}^{n}\mathbf{q}_t$, and attach a two-layer MLP to produce a predicted label $\hat{\mathbf{y}}$.

(Weakly) supervised training. Our whole model is trained in a supervised fashion, by computing the following cross-entropy loss between predicted label $\hat{\mathbf{y}}$ and ground-truth label $\mathbf{y}$

$$\mathcal{L} = -\sum_{d_i \in \mathcal{D}} p(\hat{\mathbf{y}}_i)\log p(\mathbf{y}_i). \tag{4}$$

In this way, our whole model is trained in an end-to-end fashion towards the particular downstream task, while the text encoder, concept generator and interaction generator can mutually enhance each other in a closed loop. In particularly, through the graph classifier that integrates the generated concepts and interactions, the concept generation process is regularized by the interaction generation process, while the generated interactions learn to properly organize the generated concepts. Moreover, both generation processes are weakly supervised by the downstream task loss.

3 EXPERIMENTS

In this work, we provide primitive experimental evaluations towards our proposed *doc2graph* framework. We first train *doc2graph w.r.t.* the standard document classification task in an end-to-end fashion, to show its effectiveness in improving downstream tasks. Since there is no gold ground-truth for structured document summarization in terms of concept maps, we provide various case study results to reveal the quality of our generated graphs. In the future, we aim to conduct more comprehensive evaluations over *doc2graph* with more advanced neural configurations, across a richer set of downstream tasks, as well as by recruiting both offline and online human evaluators (*e.g.*, Amazon Mechanical Turk).

Datasets. To conduct comprehensive evaluations at this stage, we use three real-world text corpora from different domains: (1) 13,081 news articles (average length 88.64) from 5 categories from *NYTimes*, (2) 21,688 paper abstracts (average length 87.27) from 6 venues from *AMiner*, and (3) 25,357 reviews (average length 71.59) with 1-5 star ratings from *Yelp*.[1]

We do not use the benchmark concept map datasets proposed in [3, 18] and other works, because they do not include downstream task labels like document classes for our model training and evaluation. Moreover, we believe there is essentially no absolute gold concept map and the only way to really evaluate its quality is through downstream tasks or careful manual checking.

[1]Data sources: (1) http://developer.nytimes.com, (2) http://www.aminer.cn/citation, (3) https://www.yelp.com/dataset

Compared Algorithms. We compare *doc2graph* to two sets of baselines described as follows:

- **Non-neural traditional methods:**
  - **AutoPhrase:** We use AutoPhrase [10], a popular phrase mining algorithm, to first find high-quality phrases from the whole corpus and use them as concepts. Then we compute interaction weights among concepts as $m_{ij} = 1 - e^{-x_{ij}}$, where x_{ij} counts how many times concepts c_i and c_j co-occur in a sentence in the corresponding document.
  - **TextRank:** In contrary to AutoPhrase, TextRank [9] first constructs a co-occurrence based interaction network among all words in the corpus and then run the PageRank algorithm to select the most 'influential' words. We use these words with the highest PageRank scores as concepts, and connect them by their original interaction links generated by TextRank.
  - **CMB-MDS:** We follow [4] to implement this state-of-the-art automatic concept-map construction pipeline, and get single-document concept maps by filtering on concepts that occur in the corresponding document.
- **Neural non-graph-generation methods:**
  - **Base:** The two-layer classification MLP is directly attached to the text embedding vector $\mathbf{h}$ output by the text encoder.
  - **Concept-only:** The classification MLP is attached to the average context embedding of all concepts $\mathbf{h}^*$ output by the concept generator.
  - **Link-only:** The classification MLP is attached to the average node embedding $\mathbf{q}^*$ output by the GCN graph encoder in the same way as *doc2graph*. However, we remove the concept generator and sample random vectors from a multinomial Gaussian distribution $\mathcal{N}(0, \mathbf{I})$ to replace $\mathbf{H}^*$ as the input node features to GCN, thus only generating links from source documents with the interaction generator.

The training complexity of all neural variants is roughly the same, although our full *doc2graph* model does include more learnable parameters that lead to slightly longer training time (*e.g.*, $\times 1.3$ compared with the base model). All models are randomly initialized and trained with standard SGD in PyTorch[2]. Full implementations of all models will be made public after the acceptance of this work.

Quantitative evaluations. We randomly split labeled data into 80% for training, 10% for validation, and 10% for testing. We run the traditional baselines on all documents and construct a graph for each document without supervision, then we train a separate GCN on the graphs constructed for the documents in the training set and test with graphs constructed for the testing set to evaluate each traditional baseline. Since the baselines construct graphs with words/phrases, we look up their pre-computed word embeddings[3] to use as input node features for GCN. We train all neural models on the training set and directly evaluate them on the testing set. We fix the size of generated graphs (n) to be 10 for all algorithms. All other hyperparameters of all algorithms are slightly tuned on the validation set. In particular, we set the dimensions of all hidden layers in the LSTM, MLP and GCN models to 100, and the numbers of layers in all GCN models to 2. We fix the batch size to 64, learning rate to 1e-3, and max epoch to 200 with patience-based early stop.

[2]https://pytorch.org/docs/stable/_modules/torch/optim/sgd.html
[3]https://nlp.stanford.edu/projects/glove/

Algorithm	NYTimes		AMiner		Yelp	
	Acc.	*ratio*	*Acc.*	*ratio*	*Acc.*	*ratio*
AutoPhrase	34.41	1.11	45.60	1.19	43.18	0.99
TextRank	83.27	1.47	59.55	1.31	48.16	1.13
CMB-MDS	76.98	1.42	45.40	1.38	50.60	1.32
Base	74.93	4.63	58.04	2.43	52.44	2.34
Concept-only	82.38	3.09	63.52	2.14	56.68	1.52
Link-only	26.04	0.99	26.04	0.99	44.78	0.98
doc2graph	**85.77**	2.91	**66.29**	1.59	**62.21**	1.48

Table 1: Doc. class. accuracy (%) and test/train loss ratio.

Table 1 shows the classification performance of compared algorithms. On *Acc.*, we observe *doc2graph* to constantly outperform all baselines by significant margins (3% to 29% over the best traditional baseline and 4% to 10% over the best neural baseline). Among the traditional baselines, *TextRank* often achieves the best performance, as selecting the informative keywords is really important for correct document classification; the other two methods focus more on entity extraction, but entity names do not necessarily indicate class labels (*e.g.*, in *NYTimes*, *Chase Center* might be a high-quality entity name, but the simple word *basketball* has a stronger indication towards the *sports* class). Among the neural variants, due to the same reason, *Concept-only* has the most competitive performance, because it can explicitly select informative words, whereas *Link-only* has the worst performance, because it totally ignores words; by considering concept interactions, *doc2graph* can further improve over *Concept-only*, demonstrating the advantage of concept maps as informative document representations. Finally, we also compute the *ratio* between testing loss and training loss for all algorithms, high values of which is a signal towards overfitting. Although all neural models except for *Link-only* can often outperform traditional algorithms due to end-to-end supervised training, they are indeed more prone to overfitting, especially on the smaller dataset of *NYTimes* where the ratios reach the highest. *doc2graph* always reaches the lowest ratios, indicating its effectiveness in leveraging concept map generation as regularization to avoid overfitting.

Case Studies. Figure 3 shows concept maps constructed by different algorithms, with meaningless concepts and weak links removed for clear visualization. In general, *AutoPhrase* can discover meaningful concepts especially in terms of phrases, but they are sometimes not properly connected, resulting in counter-factual semantics (*e.g.*, in the *NYTimes* example, based on sentence-level co-occurrence, interaction links are constructed among *"researchers"*, *"claim"* and *"soul mate"*, which does not make much sense). *TextRank* is effective in selecting word-level concepts that are important in the whole corpus (*e.g.*, *"evidence"* in *NYTimes*, *"model"* in *AMiner*, and *"service"*, *"return"* in *Yelp*), but the selected concepts are often densely connected, making the results messy and less interpretable. *CMB-MDS* is good at extracting entities and relations from multiple documents, but the single-document results are often rather sparse and less useful. *doc2graph*, by attentively selecting tokens from the source document and jointly learning interactions under weak supervision, is effective in generating both high-quality concepts and meaningful links. For instance, from the *NYTimes* news, *doc2graph* clearly distills the concise and accurate knowledge that a *"questionnaire"* mentions love *"match"* through *"mathematical"* *"algorithm"*, which is *"criticized"* by *"academic"* *"researchers"*.

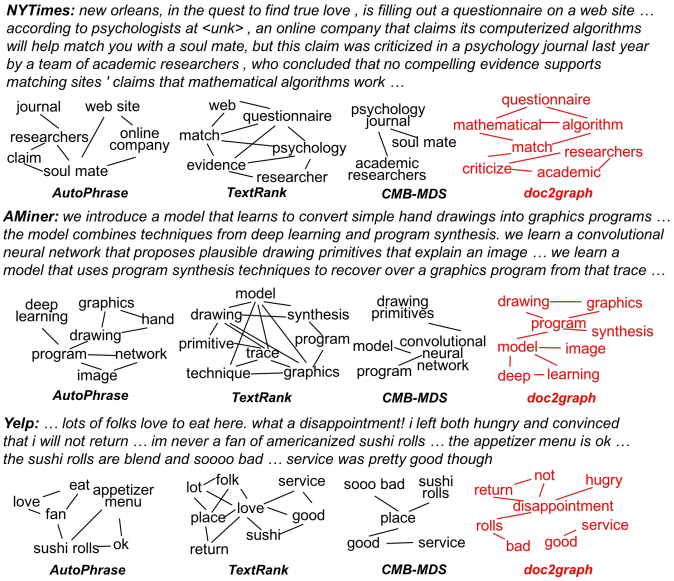

Figure 3: Case studies on three real-world text corpora.

ACKNOWLEDGEMENT

Research was sponsored in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and SocialSim Program No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and DTRA HDTRA11810026.

REFERENCES

[1] Shih-Ming Bai and Shyi-Ming Chen. 2008. Automatically constructing concept maps based on fuzzy rules for adapting learning systems. In *JESP*.
[2] MR Carvalho, Rattikorn Hewett, and Alberto J Cañas. 2001. Enhancing web searches from concept map-based knowledge models. In *WMSCI*.
[3] Tobias Falke and Iryna Gurevych. 2017. Bringing structure into summaries: Crowdsourcing a benchmark corpus of concept maps. In *EMNLP*.
[4] Tobias Falke, Christian M Meyer, and Iryna Gurevych. 2017. Concept-Map-Based Multi-Document Summarization using Concept Coreference Resolution and Global Importance Optimization. In *IJCNLP*.
[5] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[6] Raymond YK Lau and et al. 2008. Toward a fuzzy domain ontology extraction method for adaptive e-learning. In *TKDE*.
[7] Wei Li, Lei He, and Hai Zhuge. 2016. Abstractive news summarization based on event semantic link network. In *COLING*.
[8] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2018. Toward abstractive summarization using semantic representations. In *ACL*.
[9] Rada Mihalcea and et al. 2004. Textrank: Bringing order into text. In *EMNLP*.
[10] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. In *TKDE*.
[11] Yuen-Hsien Tseng and et al. 2010. Mining concept maps from news stories for measuring civic scientific literacy in media. In *JCE*.
[12] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.
[13] Oriol Vinyals and et al. 2015. Pointer networks. In *NIPS*.
[14] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *ICLR*.
[15] Carl Yang, Lingrui Gan, Zongyi Wang, Jiaming Shen, Jinfeng Xiao, and Jiawei Han. 2019. Query-Specific Knowledge Summarization with Entity Evolutionary Networks. In *CIKM*.
[16] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. 2020. Relation learning on social networks with multi-modal graph edge variational autoencoders. In *WSDM*.
[17] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. 2019. Conditional Structure Generation through Graph Variational Generative Adversarial Nets. In *NIPS*.
[18] Krunoslav Žubrinić and et al. 2015. Implementation of method for generating concept map from unstructured text in the Croatian language. In *SoftCOM*.