SemSUM: Semantic Dependency Guided Neural Abstractive Summarization

Hanqi Jin,[1,2,3*] **Tianming Wang,**[1,3*] **Xiaojun Wan**[1,2,3]
[1]Wangxuan Institute of Computer Technology, Peking University
[2]Center for Data Science, Peking University
[3]The MOE Key Laboratory of Computational Linguistics, Peking University
{jinhanqi, wangtm, wanxiaojun}@pku.edu.cn

Abstract

In neural abstractive summarization, the generated summaries often face semantic irrelevance and content deviation from the input sentences. In this work, we incorporate semantic dependency graphs about predicate-argument structure of input sentences into neural abstractive summarization for the problem. We propose a novel semantics dependency guided summarization model (SemSUM), which can leverage the information of original input texts and the corresponding semantic dependency graphs in a complementary way to guide summarization process. We evaluate our model on the English Gigaword, DUC 2004 and MSR abstractive sentence summarization datasets. Experiments show that the proposed model improves semantic relevance and reduces content deviation, and also brings significant improvements on automatic evaluation ROUGE metrics.

Introduction

Abstractive summarization aims to compress an input text into a concise, fluent summary while retaining its main idea. In this paper, we focus on abstractive sentence summarization, which involves retelling, pruning, and generation at the sentence level (Jing and McKeown 1999). Sequence-to-sequence (Seq2seq) learning has been widely used in summarization task and produced promising results (Rush, Chopra, and Weston 2015; Nallapati et al. 2016).

However, in many cases, the generated summaries still face the problem of semantic irrelevance and deviation from the input sentence, thus cannot reflect the main meaning of the original text accurately and faithfully. For example, in the summary generated by the seq2seq model in Table1, the actual subject of the verb "hoping" is "his rivals". Nevertheless, probably because the entity "federer" seems more important in the source sentence, the summarization system regards "federer" as the subject and forges the wrong match "federer hoping". Several studies have explored various ways to address the problem. Cao et al. (2018b) extracted facts from the source sentence as an auxiliary input to correct semantic errors and false facts generated by the neural network. Song, Zhao, and Liu (2018) combined source syntactic structures into neural sentence summarization to help the model identify summary-worthy content and avoid content deviation. Li et al. (2018) incorporated entailment knowledge into abstractive summarization models under a multi-task framework.

Source	even some of his rivals in the roland garros locker-room are hoping that roger federer can create a bit of tennis history by winning all four grand slams .
Target	even fellow players keen for federer grand slam dream picture
Seq2seq	federer hoping to win at roland garros
SemSUM	even some rivals hope federer to make grand slam history

Table 1: Example summaries of a sentence with and without semantics guidance.

In this paper, we aim to guide the neural summarization system with the semantic dependency graph of the source sentence. Semantic dependency graphs represent predicate-argument relations between content words in a sentence and have various semantic representation schemes (e.g., DM, PAS, PSD and CCD) based on different annotation systems (Oepen et al. 2016). Among these semantic representation schemes, DM has higher consistency and accuracy, so we leverage it as an additional input to guide summary generation. Its nodes are words and edges are labeled to encode semantic relations between the words. Non-content words, such as punctuation, are left out of the analysis. Figure 1 shows the corresponding semantic dependency graph (DM) of the source sentence in Table 1, which includes most semantically relevant local (e.g., from "some" to "rivals") and long-distance (e.g., from "hoping" to "some") dependencies. With the above semantic dependency graph incorporated, we can generate a semantically consistent summary, where "hope" finds the true subject "some rivals".

Transformer (Vaswani et al. 2017) has advanced the state-of-the-art on various translation and generation tasks. We propose a semantic dependency guided summarization model based on the Transformer, which can incorporate the semantic dependency graph and the input text by stacking

*Equal contribution.

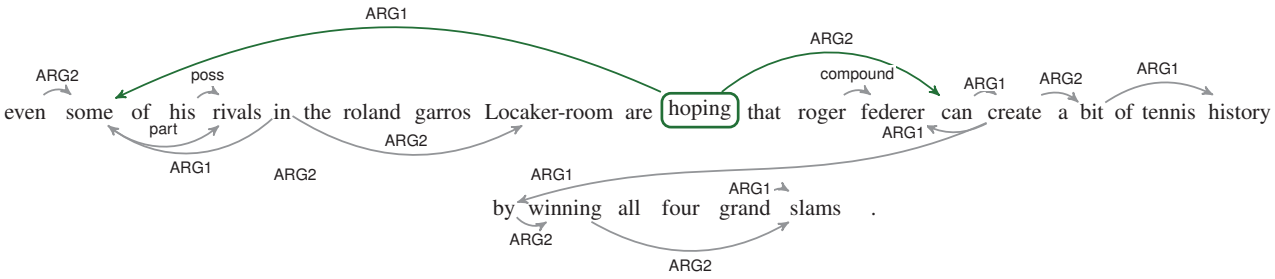

Figure 1: An example sentence annotated with a semantic dependency graphs. The green color represents the dependency of root node "hoping". Some dependency edges are omitted for display.

encoders to guide summary generation process. The stacked encoders consist of a sentence encoder and a graph encoder, which can incorporate the semantic dependency graph and the input text to generate summary in a complementary way. First, the sentence encoder reads the input sentence through stacked multi-head self-attention blocks to construct a contextual-level sentence representation. Then the graph encoder captures semantic relationships and incorporates the semantic graph structure into the contextual-level representation. The semantic dependency graph is represented as a set of relation triples (i.e., $(head, type, tail)$ if there is an edge from the sender node $head$ to the receiver node $tail$ with the label $type$). We adopt a graph attention mechanism to aggregate information of these relation triples into the corresponding sender and receiver nodes to construct a semantics-aware representation. Finally, a sentence decoder is employed for producing the output summary with attention to the semantic-aware representation.

Experiments on the Gigaword dataset show that our approach significantly improves strong baselines. We also evaluate our model on test-only DUC2004 and MSR abstractive sentence summarization datasets and it yields a large improvement.

The contributions of this work are summarized as follows:

- To the best of our knowledge, we are the first to explore semantic dependency graph for abstractive summarization. We propose a novel semantic dependency guided summarization model that leverages the input sentence and semantic dependency graph to generate summary in a complementary way. Our code is publicly available at https://github.com/zhongxia96/SemSUM.
- Our proposed model can generate summaries with high semantic relevance and readability and outperforms various baseline models on three benchmark datasets.

Related Work

The research on abstractive summarization can be divided into two categories. One class focuses on improving the architecture of the model to enhance generalization performance of summarization methods (Kikuchi et al. 2016; Zhou et al. 2017). Another type is devoted to explicitly introducing the information from other aspects to aid in the summary generation. Nallapati et al. (2016) enriched the encoder with lexical features such as named entities and POS tags. Takase et al. (2016) encoded results obtained from an abstract meaning representation (AMR) parser using a modified version of Tree-LSTM as additional information of the Attention-based Summarization model. Song, Zhao, and Liu (2018) artificially constructed features based on syntactic information and introduce it into summary generation. Cao et al. (2018b) proposed to force the generation conditioned on both the source text and the extracted fact descriptions from it. Cao et al. (2018a) and Wang, Quan, and Wang (2019a) used existing summaries as soft templates to guide the seq2seq model. Fernandes, Allamanis, and Brockschmidt (2018) incorporated name entities and coreferences with a sequence-graph model to reason about long-distance relationships. However, to our knowledge, no existing work has exploited semantic dependency graphs for enhancing neural abstractive summarization.

We fill in this gap and adopt graph neural networks as the graph encoder to leverage the knowledge with a semantic dependency graph. Graph neural network is a series of neural architectures (Scarselli et al. 2009; Gilmer et al. 2017) specifically devised to induce the representation of nodes in a graph. Marcheggiani, Bastings, and Titov (2018) encoded graph-structured data using a Graph Convolution Networks (GCN). Velickovic et al. (2018) propose a Graph Attention Networks (GAT) which leverages an attention mechanism to operate on graph-structured data and computes the hidden representations of each node in a graph by attending over its neighbors' hidden representations. It is a direct descendant of the convolutional method and offered more modeling power. Similar to the GAT, we use an attention mechanism to aggregate the neighbor relationships within semantic dependency graphs.

Problem Formulation

Given an input sentence $X = (x_1, x_2, \ldots, x_N)$, where N is the sentence length. The corresponding summary output is $Y = (y_1, y_2, \ldots, y_M)$, where $M \leq N$ is the summary length. Using existing tools, we can parse (accepting some noise) X into its semantic dependency graph $G = (V, E)$, where V is the set of nodes, and E is the set of edges. Each edge represents a semantic relation, denoted as a triple $(head, type, tail)$ and $head \in V, tail \in V, type \in R$ where $R = \{ARG1, ARG2, compound, \ldots\}$ which denotes the collection of semantic relation labels. The task can be for-

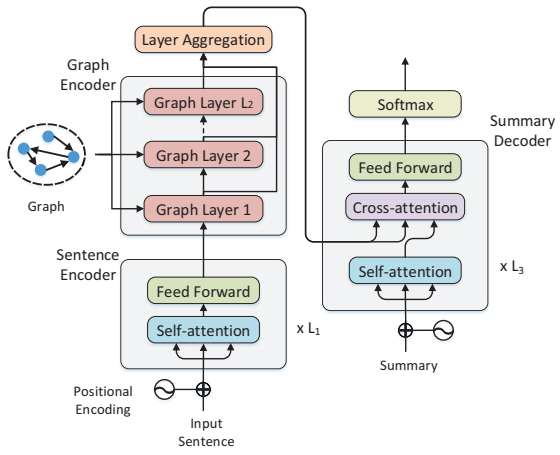

Figure 2: The overview of our SemSUM model

mally defined as: given an input sentence X and its semantic dependency graph G, the goal is to generate a target summary Y.

Our Model

Our model consists of a sentence encoder, a graph encoder and a summary decoder. Firstly, the sentence encoder reads the input sentence and builds its contextual-level representation. Then the graph encoder captures the semantic relationships according to the contextual-level representation and the semantic dependency graph to produce a semantic-aware sentence representation. Lastly, the decoder produces the output summary with attention to the semantic-aware representation. In the following sections, we introduce the sentence encoder, the graph encoder, and the summary decoder respectively.

Sentence Encoder

The role of the sentence encoder is to read the input sentence X and construct its contextual-level representation. This part is the same as (Vaswani et al. 2017), and we will give a brief introduction. The sentence encoder is a stack of L_1 identical layers. Each layer has two sub-layers: the first is a multi-head self-attention mechanism, and the second is a fully connected feed-forward network for transformation. At the bottom of the encoder stack, each input token x_i is converted into the vector representation e_{x_i} by learned embeddings. Since the Transformer is a non-recurrent model, we need to assign a "positional encoding" p_i to indicate the position of the word in the sentence, and the input representation can be obtained by simply adding these two representations: $s_i^0 = e_{x_i} + p_i$. For convenience, we denote the output of l-th layer in the sentence encoder as s^l and the input for the first layer as s^0

In multi-head attention sub-layers, the input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

$$\begin{aligned} \mathrm{Attn}(Q,K,V) &= \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \\ \mathrm{head}_j &= \mathrm{Attn}(QW_j^Q, KW_j^K, VW_j^V) \\ \mathrm{MHAtt}(Q,K,V) &= \left(\Big\|_{j=1}^{H} \mathrm{head}_j\right) W^O \end{aligned} \tag{1}$$

where $\|$ denotes the concatenation of the H attention heads, the projections are parameter matrices $W_j^Q, W_j^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_j^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $W^O \in \mathbb{R}^{H*d_v \times d_{\text{model}}}$. Specially, $Q = K = V$ in multi-head self-attention.

The feed-forward network consists of two linear transformations with a ReLU activation in between.

$$\mathrm{FFN}(x) = \max\left(0, xW_1 + b_1\right)W_2 + b_2 \tag{2}$$

To construct deep network, residual connection (He et al. 2016) and layer normalization (Ba, Kiros, and Hinton 2016) are used to connect adjacent layers.

So the whole sentence encoder works as

$$\begin{aligned} \tilde{s} &= \mathrm{LayerNorm}\left(s^{l-1} + \mathrm{MHAtt}\left(s^{l-1}, s^{l-1}, s^{l-1}\right)\right) \\ s^l &= \mathrm{LayerNorm}(\tilde{s} + \mathrm{FFN}(\tilde{s})) \end{aligned} \tag{3}$$

where $l \in [1, L_1]$, and the final contextual-level representation s^{L_1} is fed to the graph encoder.

Graph Encoder

In order to incorporate the semantic graph as an additional input, we use a graph encoder to aggregate neighbor relations for each node. The graph encoder includes two parts: graph layers and layer aggregation. Graph layers is a stack of L_2 identical layers. As shown in Figure 3, each layer has two sub-layers. The first is a graph attention mechanism, and the second is a fully connected feed-forward network. Like the sentence encoder, we also use a residual connection around each sublayer, followed by layer normalization. For convenience, we denote the output of l-th graph layer as g^l and the input of the first layer as g^0. We set the contextual-level vector $s_{pos(v)}^{L_1}$ as the initial state g_v^0 of $v \in V$ in the graph encoder, where $pos(v)$ is the position index of node v in source sentence X.

Graph attention mechanism Similar to (Velickovic et al. 2018), we use an attention mechanism to leverage the knowledge within semantic dependency graphs. The graph attention mechanism proposed in (Velickovic et al. 2018) is designed for undirected graphs. However, the relationships in semantic dependency graph is directed and have labels, so we modify it to aggregate neighbor relation triples rather than directly aggregate neighbor nodes. For simplicity and clarity, we omit the layer tag l for nodes and relations in the graph attention mechanism. For each relation $(head, type, tail)$, we first concatenate the node representations g_{head} and g_{tail} and the learnable embedding of the edge e_{type}, and then compress it through a linear transformation followed by a nonlinear activation function ReLU.

In this way, we can get the representation of each relation, which can be used in the aggregation operation of graph attention for the corresponding nodes. Considering that a given relation might have different influences on its sender node and receiver node, we use different linear transformations during compressing:

$$\begin{aligned} u_{out} &= \mathrm{ReLU}([g_{head} \parallel e_{type} \parallel g_{tail}]W_{out} + b_{out}) \\ u_{in} &= \mathrm{ReLU}([g_{head} \parallel e_{type} \parallel g_{tail}]W_{in} + b_{in}) \end{aligned} \tag{4}$$

where $W_{in}, W_{out} \in \mathbb{R}^{d_{3*\,\mathrm{model}} \times d_{\mathrm{model}}}, b_{in}, b_{out} \in \mathbb{R}^{d_{\mathrm{model}}}$. u_{out} and u_{in} are the representations of the relation, which will be aggregated by the sender node $head$ and the receiver $tail$ respectively.

The graph attention induces a new representation $\hat{g}_v$ of a node v by aggregating the representation of its outgoing neighbor relations u_{out} (v acts as a sender), and incoming neighbor relations u_{in} (v acts as a receiver).

$$\begin{aligned} \hat{g}_v &= \sum_{u \in \mathcal{N}(v)} \alpha(u, g_v) u W^V \\ \alpha(u, g_v) &= \frac{\exp\left(\left(uW^K\right)^\top g_v W^Q\right)}{\sum_{z \in \mathcal{N}(v)} \exp\left(\left(zW^K\right)^\top g_v W^Q\right)} \end{aligned} \tag{5}$$

where $\mathcal{N}_v$ denotes all neighbor relation representations of v in G, including incoming relations and outgoing relations.

Like self-attention in the sentence encoder, we also use a multi-head operation in graph attention.

$$\mathrm{MHGAT}(g_v) = \left(\Big\Vert_{j=1}^{H} \hat{g}_v^j \right) W^O \tag{6}$$

where $\hat{g}_v^j$ is the result $\hat{g}_v$ of graph attention in head j. Each head j learns independent transformations $W_j^Q, W_j^K \in \mathbb{R}^{d_{model} \times d_k}, W_j^V \in \mathbb{R}^{d_{model} \times d_v}, W^O \in \mathbb{R}^{Hd_v \times d_{\mathrm{model}}}$ respectively.

The feed-forward network and residual connections are employed to further integrate the results of multi-head graph attention. So the whole graph layers work as

$$\begin{aligned} \tilde{g} &= \mathrm{LayerNorm}\left(g^{l-1} + \mathrm{MHGAT}\left(g^{l-1}\right)\right) \\ g^l &= \mathrm{LayerNorm}\left(\tilde{g} + \mathrm{FFN}(\tilde{g})\right) \end{aligned} \tag{7}$$

Layer aggregation Through the multi-layer graph layers, the semantic information passes on among the graph nodes. A neighborhood range is expanded by performing graph attention layer by layer, and each node can cover a range of neighborhoods with a radius of L_2 when the graph layers have L_2 layers. In some parts of the graph, a sufficient neighborhood helps spreading semantic information with "long distance" dependency. However, for other nodes, a too broad expansion may also bring too much useless information and thus makes it insensitive to the information of adjacent nodes.

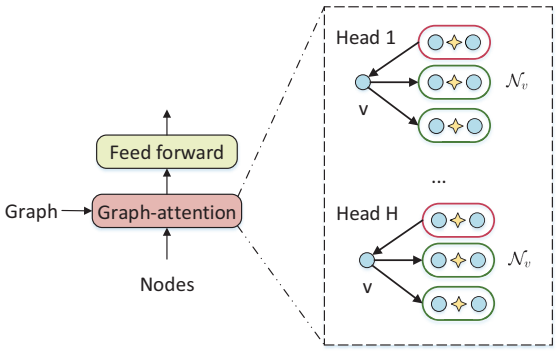

Figure 3: The graph layer consists of a graph attention mechanism and a feed-forward network. Through the graph attention, each node merges the neighbor relations. The neighbor relations are represented as triples, and the incoming relations and the outgoing relations are obtained through different mappings, which are marked in red and green color respectively in the Figure.

Following Xu et al.(2018) 's idea, we adopt a bidirectional LSTM (Hochreiter and Schmidhuber 1997) to aggregate the outputs of all graph layers. LSTM-aggregation is node adaptive where each node can independently incorporate the information from different graph layers. In other words, nodes learn to use wide-range or small-range feature through the LSTM layer aggregation. We input semantic-aware representations $g^1, \ldots, g^{L_2}$ into a bi-directional LSTM (Hochreiter and Schmidhuber 1997). The forward LSTM reads the semantic-aware representations from left to right and gets a sequence of hidden states, $\left(\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_N\right)$. The backward LSTM reads the semantic-aware representations reversely, from right to left, and results in another sequence of hidden states, $\left(\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_N\right)$. We add the last forward hidden state $\overrightarrow{h}_N$ and backward hidden state $\overleftarrow{h}_1$ as the final output o of graph encoder.

$$o = \overleftarrow{h}_1 + \overrightarrow{h}_N \tag{8}$$

Summary Decoder

The decoder is also a stack of L_3 identical sub-layers. The sub-layer consists of three parts: a masked multi-head self-attention mechanism, a multi-head cross-attention mechanism, and a fully connected feed-forward network.

Similar to the sentence encoder, we add positional encodings to the input embeddings at the bottoms of the decoder stack. We denote the output of the l-th layer as d^l and the input for the first layer as d^0.

The self-attention sub-layer with a masking mechanism is used to encode the information of decoded subsequences. The masking mechanism ensures the predictions for position t only depend on the known outputs at positions preceding t.

$$\tilde{d} = \mathrm{LayerNorm}\left(d^{l-1} + \mathrm{MHAtt}\left(d^{l-1}, d^{l-1}, d^{l-1}\right)\right) \tag{9}$$

The output of the self-attention is fed to the cross-attention sub-layer and feed-forward network. The cross-attention sub-layer performs multi-head attention over the output o of the graph encoder.

$$\begin{aligned} c &= \text{LayerNorm}\left(\tilde{d} + \text{MHAtt}\left(\tilde{d}, o, o\right)\right) \\ d^l &= \text{LayerNorm}\left(c + \text{FFN}(c)\right) \end{aligned} \tag{10}$$

The final output $d_t^{L_3}$ at the position t, is then passed through a softmax layer to generate the probability p_t^g of next word over the target vocabulary.

$$p_t^g = \text{softmax}\left(d_t^{L_3} W_g + b_g\right) \tag{11}$$

where $W_g \in \mathbb{R}^{d_{model} \times d_{vocab}}$, $b_g \in \mathbb{R}^{d_{vocab}}$ and d_{vocab} is the size of target vocabulary.

To tackle the problem of out-of-vocabulary (OOV) words, we compute the copy attention ε_t between d^{L_3} and the input representations o to obtain copy distribution p_t^c.

$$\begin{aligned} \varepsilon_t &= \text{softmax}(d_t^{L_3} o^\top + b_\varepsilon) \\ p_t^c &= \sum_{i=1}^{n} \varepsilon_t z_i^\top \end{aligned} \tag{12}$$

where z_i is the one-hot indicator vector for w_i and $b_\varepsilon \in \mathbb{R}^{d_{vocab}}$. The generation probability $\eta_t \in [0, 1]$ is calculated from the decoder output d^{L_3}.

$$\eta_t = \sigma\left(d_t^{L_3} W_\eta + b_\eta\right) \tag{13}$$

where $W_\eta \in \mathbb{R}^{d_{model} \times 1}$, $b_\eta \in \mathbb{R}^1$. The final distribution p_t is given by the "mixture" of the two probabilities with η_t.

$$p_t = \eta_t * p_t^g + (1 - \eta_t) * p_t^c \tag{14}$$

Objective Function

Given the input sentence, our goal is to maximize the probability of output summary. We use D to denote the training set, θ to represent the set of parameters. The following negative logarithm likelihood function is optimized:

$$J(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(X,Y,G) \in \mathcal{D}} \log p(Y|X, G) \tag{15}$$

Experiments

Datasets

We experiment with the English Gigaword dataset[1] (Napoles, Gormley, and Durme 2012), the DUC2004 dataset (Over, Dang, and Harman 2007) and the MSR-ATC Test Set (Toutanova et al. 2016). The Gigaword dataset contains about 3.8M sentence-summary pairs for training and 189K pairs for development. For test, we use the standard test set of 1951 sentence-summary pairs. The DUC2004 dataset has 500 input sentence with each sentence paired with 4 different human-written reference summaries. The MSR-ATC Test Set has 785 input sentences with each sentence paired with 3-5 summaries. Noted that the same model trained on Gigaword training set is evaluated on Gigaword test set, DUC2004 test set and MSR-ATC test set, respectively. We parse the source sentences of these datasets with a off-the-shelf semantic dependency parser (Chen et al. 2018), which is a neural Maximum Subgraph parser and achieves very competitive results for both English and Chinese, to get the corresponding semantic dependency graphs.

Implementation Details

We set our model parameters based on preliminary experiments on the development set. We prune the vocabulary to 50k and use the word in source sentence with maximum weights in copy attention to replace the unknown word to solve the OOVs problem. We set the dimension of word embeddings and hidden units d_{model} to 512, feed-forward units to 2048. We set 4 heads for multi-head graph-attention and 8 heads for multi-head self-attention, masked multi-head self-attention and multi-head cross-attention. We set the number of layers of sentence encoder L_1, graph encoder L_2, and summary decoder L_3 to 4, 3 and 6, respectively. We set dropout rate to 0.1 and use Adam optimizer with an initial learning rate $\alpha = 0.0001$, momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay $\epsilon = 10^{-5}$. The learning rate is halved if the valid loss on the development set increases for two consecutive epochs. We use a mini-batch size of 300. Beam search with beam size of 5 is used for decoding.

Metrics and Baselines

We use ROUGE (Lin 2004) to evaluate the generated summary in our experiments. Following previous work, we report ROUGE F1[2] on Gigaword and MSR-ATC, and ROUGE recall[3] on DUC2004. Furthermore, we evaluate the models with the latest proposed MoverScore (Zhao et al. 2019) and BertScore (Zhang et al. 2019) metrics, which correlate with human judgment much better than ROUGE. We report the Word Mover's Distance (WMD) unigram on the three datasets, which correlates with human judgement better on summarization task and has been verified in the original paper. We report the BERTScore[4] F1 on the three datasets. We compare our model with several baselines proposed in the latest years. Besides, we also implement two baselines. The first is Transformer (Vaswani et al. 2017), a multi-layer and multi-head attention architecture, which is the state-of-the-art NMT model and has been widely used in various generation tasks. We apply it for the sentence summarization task here and use the open-source toolkit fairseq (Ott et al. 2019) to train it on the Gigaword corpus. To verify the effectivess of the proposed graph encoder part, we also implement another baseline model (i.e., TFM&GCN), which keeps both the sentence encoder and summary decoder unchanged, and replace the graph encoder with the Graph Convolution Net-

[1] All the training, validation and test dataset can be downloaded at https://github.com/harvardnlp/sent-summary.

[2] The ROUGE evaluation option: -m -n 2 -w 1.2

[3] The ROUGE evaluation option: -m -b 75 -n 2 -w 1.2

[4] We use the bert-large-uncased_L18_no-idf version for the BERTScore model

works (GCN) (Marcheggiani, Bastings, and Titov 2018). We tune the best parameter configurations for the baselines.

Automatic Evaluation

Model	RG-1	RG-2	RG-L	WMD	BERT
ABS(Rush, Chopra, and Weston 2015)	29.55	11.32	26.42	-	-
SEASS(Zhou et al. 2017)	36.15	17.54	33.63	-	-
Re3Sum(Cao et al. 2018a)	37.04	19.03	34.46	-	-
FTSum(Cao et al. 2018b)	37.27	17.65	34.24	-	-
PostEnsemble(Kobayashi 2018)	37.52	18.55	34.86	-	-
Sun-Attention(Niu et al. 2019)	38.27	16.45	36.08	-	-
MASS(Song et al. 2019)	38.73	19.71	35.96	34.28	**61.56**
BiSET(Wang, Quan, and Wang 2019b)	**39.11**	**19.78**	**36.87**	33.79	61.24
Transformer	36.69	18.08	34.22	32.50	60.48
TFM&GCN	37.51	19.03	34.89	33.67	61.02
SemSUM	38.78	**19.75**	36.09	**34.39**	**61.56**

Table 2: ROUGE F1, WMD unigram and BERTScore F1 evaluation results on the Gigaword test set.

In Table 2, we report the results on the Gigaword test set. The Transformer model performs much better than ABS and achieves a 6.76 points improvement on the ROUGE-2 F1, which demonstrates the superiority of the Transformer architecture. Our model gains an improvement of 1.67 points on ROUGE-2 F1, 1.59 points on the WMD unigram and 1.89 points on the BERTScore F1 compared with Transformer, which verifies the effectiveness of semantic information for summary generation. Our model also outperforms the TFM&GCN model by 1.27 points on ROUGE-1 F1, 0.72 points on ROUGE-2 F1, 0.72 points on the MoverScore unigram (WMD) and 0.54 points on the BERTScore F1. It indicates that the proposed graph encoder in our model is better than GCN in capturing semantic information. Although BiSET obtains better scores on ROUGE metrics, our model outperforms it on both the Moverscore and BERTScore metrics and achieves a 0.32 points improvement on the BERTScore F1. The result of MASS is not strictly comparable because it uses a 190M news corpus to pre-train its language model. Our model still achieves comparable performances on the all metrics [5].

In Table 3 and Table 4, we report the results on two test-only datasets. Our proposed model SemSUM achieves the best performances among all compared models. It indicates our proposed model has a good transferability between different datasets. On DUC2004, our model achieves scores of 31.00, 11.11 and 26.94 on three ROUGE metrics respectively, which are about 1 points higher than the scores of Transformer. On MSR-ATC, the gap between SemSUM and Transformer goes up to more than 4. These result verify the effectiveness of semantic dependency information again. In addition, our model still has a better performance than TFM&GCN.

In general, our model achieves strong performances on three benchmark datasets and shows its effectiveness and stability on the summarization task.

[5]To calculate the MoverScore and BERTScore, we request the BiSET outputs from the authors and get the MASS outputs via running the trained model which is uploaded by the authors to Github.

Model	RG-1	RG-2	RG-L	WMD	BERT
ABS(Rush, Chopra, and Weston 2015)	26.55	7.06	22.05	-	-
SEASS(Zhou et al. 2017)	29.21	9.56	25.51	-	-
ERAML(Li et al. 2018)	29.33	10.24	25.24	-	-
WACNNs(Yuan et al. 2019)	30.54	10.87	26.94	-	-
Transformer	29.78	9.61	25.85	22.95	56.36
TFM&GCN	30.24	10.44	26.32	24.80	57.21
SemSUM	**31.00**	**11.11**	**26.94**	**26.71**	**57.99**

Table 3: ROUGE recall, WMD unigram and BERTScore F1 evaluation results on the DUC2004 test set.

Model	RG-1	RG-2	RG-L	WMD	BERT
ABS(Rush, Chopra, and Weston 2015)	20.27	5.26	17.10	-	-
SEASS(Zhou et al. 2017)	25.75	10.63	22.90	-	-
Transformer	29.29	12.45	25.93	13.76	54.31
TFM&GCN	32.53	15.41	29.26	15.61	55.48
SemSUM	**33.82**	**17.08**	**30.62**	**17.14**	**56.19**

Table 4: ROUGE F1, WMD unigram and BERTScore F1 evaluation results on the MSR-ATC test set.

Human Evaluation

To further verify whether our model can improve semantic relevance and reduce content deviation, we carry out a human evaluation. We focus on three aspects: **faithfulness**, **informativeness**, and **fluency**. The faithfulness indicator directly measures the semantic relevance and faithfulness to the original input; the informativeness indicator can reflect whether there is a content deviation; the fluency focuses on the quality of the language. We sample 100 instances from the Gigaword test set and employ 6 graduate students to rate each summary. 3 human judgments are obtained for every sample and the final scores are averaged across different judges.

Results are presented in Figure 4. We can clearly see that our model not only performs much better than Transformer and TFM&GCN, but also achieves a comparable performance as the ground truth summaries. In the faithfulness indicator, SemSUM outperforms the Transformer baseline by a large margin, which indicates the use of the semantic dependency graph does help improving the semantic relevance of the generated summary. In the informativeness indicator, our model achieves a high score of 3.8, which is higher than 3.27 of Transformer and 3.51 of GCN, and is

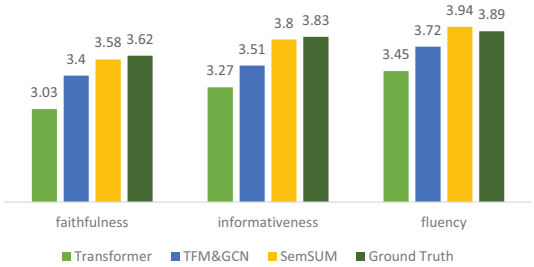

Figure 4: Human evaluation. They are rated on a Likert scale of 1(worst) to 5(best).

Model	RG-1	RG-2	RG-L
SemSUM	48.25	26.54	45.14
only sentence encoder	47.14	24.94	44.00
only graph encoder	47.28	25.21	44.17
only graph encoder*	47.50	25.51	44.39
without layer aggregation	48.02	25.83	44.82
without separated mappings	47.82	25.57	44.59

Table 5: ROUGE F1 evaluation results on the development set of ablation study. $\star$ denotes taking the adjacent relation as an edge type.

close to 3.83 of the ground truth. It indicates that our model can effectively reduce the content deviation phenomenon. In the fluency indicator, our model is 0.49 better than Transformer on ROUGE-2 F1, indicating that our model can reduce the grammatical mismatch and improve the readability of the summary. In all three aspects, our model outperforms TF&GCN method, which shows that our model has an advantage over TF&GCN method in incorporating the semantic dependency graph into summarization models.

Ablation Study

We perform ablation study on the development set to investigate the influence of different modules in our proposed SemSUM model. Modules are tested in five ways: (1) we remove the graph encoder to verify the effectiveness of semantic relations; (2) we remove the sentence encoder part and only encode the semantic dependency graph; (3) we remove the sentence encoder and add position-adjacent relations as edges in dependency graph; (4) we remove the layer aggregation and use only the output of the last graph layer as the final output of the graph encoder; (5) we remove the separated mappings (W_{in}, b_{in} and W_{out}, b_{out}) in different directions and use only one set of parameters (W and b) .

Table 5 presents the results. The best hyperparameter configuration is chosen for each model. We found that the ROUGE-2 F1 score drops by 1.6 when the graph encoder is removed. ROUGE-2 F1 score drops by 1.33 after the sentence encoder is removed. Treating position-adjacent relation as an edge type improves the ROUGE-2 F1 score by 0.3, but the improvement is limited. It indicates learning a contextual-level representation through sentence encoder is beneficial to the encoding of input sentences. ROUGE-2 F1 score drops by 0.71 after layer aggregation is removed, which shows that explicitly aggregating different levels of semantic dependency propagation can improve the performance of the model. ROUGE-2 F1 score drops by 0.97 after the separated mappings are removed. It indicates distinguishing the mapping in different directions is necessary to improve the performance of the model.

Case Study

We perform case studies for better understanding the model performances. In Table 6, we show two example outputs of the Transformer and SemSUM. In the first case, the Transformer model distorts the facts of the input text. The original expression means "call on ioc to do more for the women", while the baseline model completely ignores the structure of the sentence, only focuses on "women", produces "call for more women". In contrast, SemSUM correctly summarizes the original content "urges ioc to do more for the women". The reason for verb substitution should be that the "call on" is often replaced with "urge" in the training data. In the second case, the baseline summary is ungrammatical and not smooth. We infer that the distance between "democrats" and "promoting" is so far that the baseline model fails to judge the real subject of "promote". But our model can capture the semantic relation between "democrats" and "promoting" and generates a better summary.

Source	german parliament called on the international olympic committee on thursday to do more for women in sport .
Target	olympics told to help women
Transformer	german parliament calls for more women in sport
SemSUM	german parliament urges ioc to do more for women
Source	democrats in georgia and alabama , borrowing an idea usually advanced by conservative republicans , are promoting bible classes in the public schools .
Target	democrats in southern states push bills on bible study
Transformer	bible classes in public schools are promoting bible classes
SemSUM	democrats promote bible classes in public schools

Table 6: Case Study.

Conclusion and Future Work

In this paper, we explore incorporating semantic dependency graphs into abstractive summarization models. We propose a novel model SemSUM, which can leverage the information of original input texts and corresponding semantic dependency graphs to guide summarization process, and our model achieves strong performances on three datasets. Both automatic evaluation and human evaluation indicate that our proposed model improves semantic relevance, resulting in higher-quality summaries.

In the future, we will incorporate more knowledge like Knowledge Graph and Abstract Meaning Representation to further improve the performance.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Tencent AI Lab Rhino-Bird Focused Research Program (No.JR201953) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Xiaojun Wan is the corresponding author.

References

Ba, L. J.; Kiros, R.; and Hinton, G. E. 2016. Layer normalization. *CoRR* abs/1607.06450.

Cao, Z.; Li, W.; Li, S.; and Wei, F. 2018a. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*.

Cao, Z.; Wei, F.; Li, W.; and Li, S. 2018b. Faithful to the original: Fact aware neural abstractive summarization. In *AAAI*, 4784–4791.

Chen, Y.; Huang, S.; Wang, F.; Cao, J.; Sun, W.; and Wan, X. 2018. Neural maximum subgraph parsing for cross-domain semantic dependency analysis. In *CoNLL 2018*, 562–572.

Fernandes, P.; Allamanis, M.; and Brockschmidt, M. 2018. Structured neural summarization. *CoRR* abs/1811.01824.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8).

Jing, H., and McKeown, K. R. 1999. The decomposition of human-written summary sentences. In *SIGIR*.

Kikuchi, Y.; Neubig, G.; Sasano, R.; Takamura, H.; and Okumura, M. 2016. Controlling output length in neural encoder-decoders. In *EMNLP*, 1328–1338.

Kobayashi, H. 2018. Frustratingly easy model ensemble for abstractive summarization. In *EMNLP*.

Li, H.; Zhu, J.; Zhang, J.; and Zong, C. 2018. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *COLING*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Marcheggiani, D.; Bastings, J.; and Titov, I. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *NAACL-HLT 2018*, 486–492.

Nallapati, R.; Zhou, B.; dos Santos, C. N.; Gülçehre, Ç.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL 2016*.

Napoles, C.; Gormley, M. R.; and Durme, B. V. 2012. Annotated gigaword. In *AKBC-WEKEX@NAACL-HLT 2012*.

Niu, J.; Sun, M.; Rodrigues, J. J.; and Liu, X. 2019. A novel attention mechanism considering decoder input for abstractive text summarization. In *ICC*. IEEE.

Oepen, S.; Kuhlmann, M.; Miyao, Y.; Zeman, D.; Cinková, S.; Flickinger, D.; Hajic, J.; Ivanova, A.; and Uresová, Z. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *LREC 2016*.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Over, P.; Dang, H.; and Harman, D. 2007. DUC in context. *Inf. Process. Manage.* 43(6).

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *IEEE Trans. Neural Networks* 20(1).

Song, K.; Tan, X.; Qin, T.; Lu, J.; and Liu, T. 2019. MASS: masked sequence to sequence pre-training for language generation. In *ICML*.

Song, K.; Zhao, L.; and Liu, F. 2018. Structure-infused copy mechanisms for abstractive summarization. In *COLING*, 1717–1729.

Takase, S.; Suzuki, J.; Okazaki, N.; Hirao, T.; and Nagata, M. 2016. Neural headline generation on abstract meaning representation. In *EMNLP*.

Toutanova, K.; Brockett, C.; Tran, K. M.; and Amershi, S. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *EMNLP*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 6000–6010.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

Wang, K.; Quan, X.; and Wang, R. 2019a. Biset: Bi-directional selective encoding with template for abstractive summarization. In *ACL*.

Wang, K.; Quan, X.; and Wang, R. 2019b. BiSET: Bi-directional selective encoding with template for abstractive summarization. In *ACL*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

Yuan, C.; Bao, Z.; Sanderson, M.; and Tang, Y. 2019. Incorporating word attention with convolutional neural networks for abstractive summarization. *World Wide Web*.

Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2019. Bertscore: Evaluating text generation with BERT. *CoRR* abs/1904.09675.

Zhao, W.; Peyrard, M.; Liu, F.; Gao, Y.; Meyer, C. M.; and Eger, S. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *EMNLP*, 563–578. Hong Kong, China: Association for Computational Linguistics.

Zhou, Q.; Yang, N.; Wei, F.; and Zhou, M. 2017. Selective encoding for abstractive sentence summarization. In *ACL*, 1095–1104.