Graph4NLP: A Library for Deep Learning on Graphs for NLP

Yu (Hugo) Chen
Research Scientist at Meta AI

DLG4NLP@ICLR’22
April 29th, 2022
Graph4NLP: A Brief History and Future

Year September/2021
Our DLG4NLP website launched: survey, library, tutorial, workshop and many more.

Year June/2021
Graph4NLP v0.4.1 Released, first library for promoting easy use of GNN for NLP.

Year September/2021
Graph4NLP v0.5.1 released, Lint the codes, support testing with users’ own data, and fix many reported bugs.

Year 01/20/2022
Graph4NLP v0.5.5 released, Support model.predict API by introducing wrapper functions, separate graph topology and graph embedding and many more...

Year May/2022 (planning)
- Graph4NLP v0.6 will be released, new configuration system, relational GNN, AMR graph construction support.

Year 2022+
- More Graph4NLP releases
- More workshops
- Release a new Graph4NLP book by Cambridge Press.
Overall Architecture of Graph4NLP Library

Key Features

**Easy-to-use and Flexible**
Provides both full implementations of state-of-the-art models and also flexible interfaces to build customized models with whole-pipeline support

**Rich Set of Learning Resources**
Provide a variety of learning materials including code demos, code documentations, research tutorials and videos, and paper survey

**High Running Efficiency and Extensibility**
Build upon highly-optimized runtime libraries including DGL and provide highly modulization blocks

**Comprehensive Code Examples**
Provide a comprehensive collection of NLP applications and the corresponding code examples for quick-start

Documentation website: [https://graph4ai.github.io/graph4nlp/index.html](https://graph4ai.github.io/graph4nlp/index.html)
Data Flow of Graph4NLP

- **Raw Data**
- **Graph Construction**
- **Encoded Structured Data** (Graph4NLP.GraphData)
- **Prediction**
- **Results**
  - **Evaluation**
  - **Loss**

- **GNN Embedding Methods**
- **Featured Structured Data** (Graph4NLP.GraphData)
  - **User Model**
Computing Flow of Graph4NLP

Text Sequence → Static Graph Construction → Graph Data: Topology Construction → Graph Data: Embedding Initialization → Dynamic Graph Construction → Node/Graph Embedding

Natural Graph in NLP

Node/Link/Graph Prediction

With mechanisms like attention, copy, coverage, etc.

Sequence/Tree/Graph Decoder
### Performance of Built-in NLP Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>GNN Model</th>
<th>Graph construction</th>
<th>Evaluation</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text classification</td>
<td>TRECT</td>
<td>GAT</td>
<td>Dependency</td>
<td>Accuracy</td>
<td>0.948</td>
</tr>
<tr>
<td></td>
<td>CAirline</td>
<td></td>
<td></td>
<td></td>
<td>0.769</td>
</tr>
<tr>
<td></td>
<td>CNSST</td>
<td></td>
<td></td>
<td></td>
<td>0.538</td>
</tr>
<tr>
<td>Semantic Parsing</td>
<td>JOBS</td>
<td>SAGE</td>
<td>Constituency</td>
<td>Execution accuracy</td>
<td>0.936</td>
</tr>
<tr>
<td>Question generation</td>
<td>SQuAD</td>
<td>GGNN</td>
<td>Dependency</td>
<td>BLEU-4</td>
<td>0.15175</td>
</tr>
<tr>
<td>Machine translation</td>
<td>IWSLT14</td>
<td>GCN</td>
<td>Dynamic</td>
<td>BLEU-4</td>
<td>0.3212</td>
</tr>
<tr>
<td>Summarization</td>
<td>CNN(30k)</td>
<td>GCN</td>
<td>Dependency</td>
<td>ROUGE-1</td>
<td>26.4</td>
</tr>
<tr>
<td>Knowledge graph completion</td>
<td>Kinship</td>
<td>GCN</td>
<td>Dependency</td>
<td>MRR</td>
<td>82.4</td>
</tr>
<tr>
<td>Math word problem</td>
<td>MAWPS</td>
<td>SAGE</td>
<td>Dynamic</td>
<td>Solution accuracy</td>
<td>76.4</td>
</tr>
<tr>
<td></td>
<td>MATHQA</td>
<td></td>
<td></td>
<td>Exact match</td>
<td>61.07</td>
</tr>
</tbody>
</table>
Dive Into Graph4NLP Library

**Modules**

- Static Graph Construction
- Dynamic Graph Construction
- Node Embedding Based
- Node Embedding Based Refined

**Prediction**

- Classification
- Link Prediction
- KG Completion

**Generation**

- Sequence Decoder
- Tree Decoder

**Loss**

**Evaluation**

**Utils**

**Dataset**

**Applications**

- Text Classification
- Semantic Parsing
- Machine Translation
- KG Completion
- Question Generation
- Math Word Problem
- Relation Extraction
- Named Entity Recognition
- Text Summarization

**Models**

- Graph2Seq
- Graph2Tree
- Graph2Graph

**Graph4NLP Library**

**Graph Construction**

- Dependency Graph
- Constituency Graph
- IE Graph

**Graph Embedding Initialization**

- GCN
- GraphSAGE
- GAT
- GGNN

**Graph Embedding Learning**

**Classification**

**Link Prediction**

**KG Completion**

**Sequence Decoder**

**Tree Decoder**
Graph Construction Module

- Static graph construction
  - Dependency graph construction
  - Constituency graph construction
  - IE graph construction

- Dynamic graph construction
  - Node embedding based
  - Node embedding based refined (i.e., static & dynamic hybrid)
Graph Embedding Initialization Module

- Single-token & multi-token node/edge
- Various built-in strategies for node/edge embedding initialization (non-exhaustive list)
  - 'w2v'
  - 'w2v_bilstm'
  - 'bert'
  - 'bert_bilstm'
  - 'w2v_bert'
  - 'w2v_bert_bilstm'

```python
self.graph_initializer = GraphEmbeddingInitialization(
    word_vocab=self.vocab_model.in_word_vocab,
    embedding_style=embedding_style,
    hidden_size=config["num_hidden"],
    word_dropout=config["word_dropout"],
    rnn_dropout=config["rnn_dropout"],
    fix_word_emb=not config["no_fix_word_emb"],
    fix_bert_emb=not config.get("no_fix_bert_emb", False),
)
```

```python
embedding_style = {
    "single_token_item": True if self.graph_name != "ie" else False,
    "emb_strategy": config.get("emb_strategy", "w2v_bilstm"),
    "num_rnn_layers": 1,
    "bert_model_name": config.get("bert_model_name", "bert-base-uncased"),
    "bert_lower_case": True,
}
```
Graph Embedding Learning Module

- Common GNN variants
  - GCN
  - GAT
  - GraphSAGE
  - GGNN
- direction_option
  - ‘undirected’
  - ‘bi_fuse’
  - ‘bi_sep’
- use_edge_weight
  - useful for dynamic graph construction

```python
self.gnn = GGNN(  
    config["gnn_num_layers"],  
    config["num_hidden"],  
    config["num_hidden"],  
    config["num_hidden"],  
    feat_drop=config["gnn_dropout"],  
    direction_option=config["gnn_direction_option"],  
    bias=True,  
    use_edge_weight=use_edge_weight,  
)```
Prediction Module

• Classification
  • Node classification
  • Graph classification
  • Link prediction
  • KG completion
  • Graph pooling: avg_pool, max_pool

• Generation
  • Sequence decoder
  • Tree decoder
  • Attention, copy, coverage mechanisms

```python
self.seq_decoder = StdRNNDecoder(
    rnn_type=rnn_type,
    max_decoder_step=decoder_length,
    input_size=input_size,
    hidden_size=hidden_size,
    graph_pooling_strategy=graph_pooling_strategy,
    word_emb=self.dec_word_emb,
    vocab=vocab_model.out_word_vocab,
    attention_type=attention_type,
    fuse_strategy=fuse_strategy,
    node_type_num=node_type_num,
    rnn_emb_input_size=rnn_input_size,
    use_coverage=use_coverage,
    use_copy=use_copy,
    tgt_emb_as_output_layer=tgt_emb_as_output_layer,
    dropout=rnn_dropout,
)
```

Built-in high-level Graph2Seq, Graph2Tree APIs. Config in, model out.
Dataset

- Built-in dataset types
  - Text2TextDataset
  - TextToTreeDataset
  - Text2LabelDataset
  - SequenceLabelingDataset
  - DoubleText2TextDataset

```python
class TrecDataset(Text2LabelDataset):
    @property
def raw_file_names(self):
        return {'train': 'train.txt', 'test': 'test.txt'}

    @property
def processed_file_names(self):
        return {'vocab': 'vocab.txt', 'data': 'data.pt', 'label': 'label.pt'}

def __init__(self, config):
    root_dir=self.config['graph_construction_args']['root_dir'],
    topology_subdir=topology_subdir,
    graph_name=self.graph_name,
    dynamic_init_graph_name=self.config['graph_construction_args']['graph_name'],
    dynamic_init_topology_aux_args={"dummy_param": 0},
    pretrained_word_emb_name=self.config['pretrained_word_emb_name'],
    merge_strategy=self.config['graph_construction_args']['merge_strategy'],
    edge_strategy=self.config['graph_construction_args']['edge_strategy'],
    min_word_vocab_freq=self.config.get('min_word_freq', 1),
    word_emb_size=self.config.get('word_emb_size', 300),
    seed=self.config['seed'],
    thread_number=self.config['thread_number'],
    port=self.config['port'],
    timeout=self.config['timeout'],
    reused_label_model=None,
```

```python
dataset = TrecDataset(
    root_dir=self.config['graph_construction_args']['root_dir'],
    topology_subdir=topology_subdir,
    graph_name=self.graph_name,
    dynamic_init_graph_name=self.config['graph_construction_args']['graph_name'],
    dynamic_init_topology_aux_args={"dummy_param": 0},
    pretrained_word_emb_name=self.config['pretrained_word_emb_name'],
    merge_strategy=self.config['graph_construction_args']['merge_strategy'],
    edge_strategy=self.config['graph_construction_args']['edge_strategy'],
    min_word_vocab_freq=self.config.get('min_word_freq', 1),
    word_emb_size=self.config.get('word_emb_size', 300),
    seed=self.config['seed'],
    thread_number=self.config['thread_number'],
    port=self.config['port'],
    timeout=self.config['timeout'],
    reused_label_model=None,
)```
Inference

- Inference wrapper
  - classifier_inference_wrapper
  - generator_inference_wrapper
  - generator_inference_wrapper_for_tree

```python
self.inference_tool = GeneratorInferenceWrapper(
    cfg=self.config, model=self.model,
    dataset=DoubleText2TextDataset,
    data_item=DoubleText2TextDataItem,
    beam_size=self.config["beam_size"],
    topk=1, lower_case=True,
    tokenizer=word_tokenize,
    share_vocab=True,
)
```

```python
self.inference_tool = ClassifierInferenceWrapper(
    cfg=self.config,
    model=self.model,
    label_names=self.model.label_model.le.classes_.tolist(),
    dataset=Text2LabelDataset,
    data_item=Text2LabelDataItem,
    lower_case=True,
    tokenizer=word_tokenize,
)
```
Ref: https://mentorphile.com/2018/09/14/demo-or-die/
Demo 1: Text Classification Application

1) git clone https://github.com/graph4ai/graph4nlp_demo
2) follow Get Started instructions in README
Demo 1: Building a Text Classification Application

```python
def forward(self, graph_list, tgt=None, require_loss=True):
    # graph embedding initialization
    batch_gd = self.graph_initializer(graph_list)

    # run dynamic graph construction if turned on
    if hasattr(self, "graph_topology") and hasattr(self.graph_topology, "dynamic_topology"):
        batch_gd = self.graph_topology.dynamic_topology(batch_gd)

    # run GNN
    self.gnn(batch_gd)

    # run graph classifier
    self.clf(batch_gd)
    logits = batch_gd.graph_attributes["logits"]

    if require_loss:
        loss = self.loss(logits, tgt)
        return logits, loss
    else:
        return logits
```

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo
Demo 1: Building a Text Classification Application

```python
embedding_style = {
    "single_token_item": True if self.graph_name != "ie" else False,
    "emb_strategy": config.get("emb_strategy", "w2v_bilstm"),
    "num_rnn_layers": 1,
    "bert_model_name": config.get("bert_model_name", "bert-base-uncased"),
    "bert_lower_case": True,
}

self.graph_initializer = GraphEmbeddingInitialization(
    word_vocab=self.vocab_model.in_word_vocab,
    embedding_style=embedding_style,
    hidden_size=config["num_hidden"],
    word_dropout=config["word_dropout"],
    rnn_dropout=config["rnn_dropout"],
    fix_word_emb=not config["no_fix_word_emb"],
    fix_bert_emb=not config.get("no_fix_bert_emb", False),
)
```

Graph embedding initialization API, various built-in options, can be customized

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo
Demo 1: Building a Text Classification Application

```python
self.graph_topology = NodeEmbeddingBasedGraphConstruction(
    sim_metric_type=config["gl_metric_type"],
    num_heads=config["gl_num_heads"],
    top_k_neigh=config["gl_top_k"],
    epsilon_neigh=config["gl_epsilon"],
    smoothness_ratio=config["gl_smoothness_ratio"],
    connectivity_ratio=config["gl_connectivity_ratio"],
    sparsity_ratio=config["gl_sparsity_ratio"],
    input_size=config["num_hidden"],
    hidden_size=config["gl_num_hidden"],
)
```

Graph construction API, various built-in options, can be customized

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo
Demo 1: Building a Text Classification Application

Graph embedding learning API, various built-in options, can be customized

```
self.gnn = GraphSAGE(config['gnn_num_layers'],
    config['num_hidden'],
    config['num_hidden'],
    config['num_hidden'],
    config['graphsage_aggreagte_type'],
    direction_option=config['gnn_direction_option'],
    feat_drop=config['gnn_dropout'],
    bias=True,
    norm=None,
    activation=F.relu,
    use_edge_weight=use_edge_weight)
```

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo
Demo 1: Building a Text Classification Application

```python
self.clf = FeedForwardNN(2 * config['num_hidden'] \ 
  if config['gnn_direction_option'] == 'bi_sep' \ 
  else config['num_hidden'],
  config['num_classes'],
  [config['num_hidden']],
  graph_pool_type=config['graph_pooling'],
  dim=config['num_hidden'],
  use_linear_proj=config['max_pool_linear_proj'])
```

Prediction API, various built-in options, can be customized

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo
Demo 1: Building a Text Classification Application

dataset = TrecDataset(
    root_dir=self.config["graph_construction_args"]['graph_construction_share']['root_dir'],
    topology_subdir=topology_subdir,
    graph_name=self.graph_name,
    dynamic_init_graph_name=self.config["graph_construction_args"]['graph_construction_private']
    "dynamic_init_graph_name"],
    dynamic_init_topology_aux_args={'dummy_param': 0},
    pretrained_word_emb_name=self.config["pretrained_word_emb_name"],
    merge_strategy=self.config["graph_construction_args"]['graph_construction_private']
    "merge_strategy"
),
edge_strategy=self.config["graph_construction_args"]['graph_construction_private']
    "edge_strategy"
),
min_word_vocab_freq=self.config.get("min_word_freq", 1),
word_emb_size=self.config.get("word_emb_size", 300),
seed=self.config["seed"],
thread_number=self.config["graph_construction_args"]['graph_construction_share']
    "thread_number"
),
port=self.config["graph_construction_args"]['graph_construction_share']
    "port"],
timeout=self.config["graph_construction_args"]['graph_construction_share']
    "timeout"],
reused_label_model=None,
)

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo

Dataset API, various built-in options, can be customized
Demo 2: Building a Math Word Problem Application

1) git clone https://github.com/graph4ai/graph4nlp_demo
2) follow Get Started instructions in README
Demo 2: Building a Math Word Problem Application

A bee has 2 legs ...... ?

X = N
2 * 1

Graph2Tree problem
Demo 2: Building a Math Word Problem Application

def _build_model(self):
    '''For encoder-decoder'''
    self.model = Graph2Tree.from_args(self.opt,
                                       vocab_model=self.vocab_model)
    self.model.init(self.opt["init_weight"])  
    self.model.to(self.device)

https://github.com/graph4ai/graph4nlp_demo/tree/main/DLG4NLP%40ICLR2022_demo/Math-word-problem
Demo 2: Building a Math Word Problem Application

Dataset API, various built-in options, can be customized

```python
para_dic = {
    "root_dir": self.data_dir,
    "word_emb_size": self.opt["graph_initialization_args"]["input_size"],
    "topology_subdir": self.opt["graph_construction_args"]["graph_construction_share"][
        "topology_subdir"
    ],
    "edge_strategy": self.opt["graph_construction_args"]["graph_construction_private"][
        "edge_strategy"
    ],
    "graph_name": self.opt["graph_construction_args"]["graph_construction_share"][
        "graph_name"
    ],
    "share_vocab": self.use_share_vocab,
    "enc_emb_size": self.opt["graph_initialization_args"]["input_size"],
    "dec_emb_size": self.opt["decoder_args"]["rnn_decoder_share"]["input_size"],
    "dynamic_init_graph_name": self.opt["graph_construction_args"][
        "graph_construction_private"
    ].get("dynamic_init_graph_name", None),
    "min_word_vocab_freq": self.opt["min_freq"],
    "pretrained_word_emb_name": self.opt["pretrained_word_emb_name"]
}
dataset = MawpsDatasetForTree(**para_dic)
```
Future Directions

- Customization
- Scalability – native multi-GPU/node training support
- Easy deployment to production
- Benchmarking – more SOTA models and NLP tasks
- TensorFlow support
- …

PRs and suggestions are welcome 😊
Resources

• DLG4NLP website: https://dlg4nlp.github.io/index.html
• Graph4NLP library: https://github.com/graph4ai/graph4nlp
• Graph4NLP documentation https://graph4ai.github.io/graph4nlp/
• Survey: https://arxiv.org/abs/2106.06090
• Literature list: https://github.com/graph4ai/graph4nlp_literature
Team Members

and Jing Hu
Thanks!

Q&A

Yu (Hugo) Chen
Research Scientist
Meta AI,
Email: hugochan2013@gmail.com, hugochen@fb.com
Web: http://academic.hugochan.net