Graph4NLP Library and Demo

Yu (Hugo) Chen
Research Scientist at Meta AI

Joint Work with Graph4NLP Team

CLIQ-ai November Meetup
Nov 19th, 2021
Graph4NLP: A Library for Deep Learning on Graphs for NLP
Overall Architecture of Graph4NLP Library

Graph4NLP: [https://github.com/graph4ai/graph4nlp](https://github.com/graph4ai/graph4nlp), DGL: [https://github.com/dmlc/dgl](https://github.com/dmlc/dgl), Huggingface Transformers: [https://github.com/huggingface/transformers](https://github.com/huggingface/transformers)
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.

DLG4NLP website: https://dlg4nlp.github.io, Graph4NLP documentation: https://graph4ai.github.io/graph4nlp/
Data Flow of Graph4NLP

1. Raw Data
2. Graph Construction
3. Featured Structured Data (Graph4NLP.GraphData)
4. GNN Embedding Methods
5. Prediction
6. Results
7. Evaluation
8. Loss
Computing Flow of Graph4NLP

Text Sequence → Static Graph Construction → Graph Data
  i. Topology construction
  ii. Embedding construction → Dynamic Graph Construction → Node/Graph Embedding

With mechanisms like attention, copy, coverage, etc.

Natural Graph in NLP → Node/Link/Graph Prediction

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
Graph Construction Module

- Topology construction
  - 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)

- Embedding construction (i.e., initialization)
  - 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_topology = DependencyBasedGraphConstruction(
    embedding_style=embedding_style,
    vocab=vocab.in_word_vocab,
    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 Module

- Common GNN variants
  - GCN
  - GAT
  - GraphSAGE
  - GGNN
- direction_option
  - ‘undirected’
  - ‘bi_fuse’
  - ‘bi_sep’
- use_edge_weight

```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
  - 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 TreeDataset(Text2LabelDataset):
    @property
    def raw_file_names(self):
        """3 reserved keys: 'train', 'val' (optional), 'test'. Represent the split of dataset."
        return {'train': "train.txt", 'test': "test.txt"}

    @property
    def processed_file_names(self):
        """At least 3 reserved keys should be filled: 'vocab', 'data' and 'label'."
        return {'vocab': "vocab.pt", 'data': "data.pt", 'label': "\label.pt"}

    def __init__(
```
Inference

- Inference wrapper
  - classifier_inference_wrapper.py
  - generator_inference_wrapper.py

```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,
)
```

```python
self.inference_tool = GeneratorInferenceWrapper(
    cfg=self.opt, model=self.model,
    beam_size=3, lower_case=True,
    tokenizer=word_tokenize
)
```
Demo 1: Building a 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

def forward(self, graph_list, tgt=None, require_loss=True):
    # build graph topology
    batch_gd = self.graph_topology(graph_list)

    # run GNN encoder
    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

Demo 1: Building a Text Classification Application

```python
self.graph_topology = DependencyBasedGraphConstruction(
    embedding_style=embedding_style,
    vocab=vocab.in_word_vocab,
    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 construction API, various built-in options, can be customized

Demo 1: Building a Text Classification Application

```python
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)
```

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

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

Demo 1: Building a Text Classification Application

```python
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",
)
```

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

Demo 2: Building a Semantic Parsing Application

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

```python
def _build_model(self):
    self.model = Graph2Seq.from_args(self.opt, self.vocacy).to(self.device)
```

Resources

• Our Graph4NLP library aims to make easy use of GNNs for NLP:
  • DLG4NLP website: https://dlg4nlp.github.io/index.html
  • Survey: https://arxiv.org/abs/2106.06090
  • Graph4NLP library: https://github.com/graph4ai/graph4nlp
  • Graph4NLP documentation https://graph4ai.github.io/graph4nlp/
  • Literature list: https://github.com/graph4ai/graph4nlp_literature
Thanks!

Q&A

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