Neural Monkey: A Natural Language Processing Toolkit

Jindřich Helcl, Jindřich Libovický, Tom Kocmi, Dušan Variš, Tomáš Musil, Ondřej Cífka, Ondřej Bojar

March 7, 2019
NLP Toolkit Overview

Why do we need NLP toolkits?

- No need to re-implement everything from scratch.
- Re-use of published (trained) components.
- Often difficult design decisions already made.
- Usually, published results indicate the reliability of the toolkit.
NLP Toolkit Overview

NLP research libraries and toolkits can be categorized as:

- **Math libraries** (matrix- or tensor-level, usually symbolic)

- **Neural Network abstraction APIs** (handle individual NN layers)
  - Keras

- **Higher-level toolkits** (work with encoders, decoders, etc.)
  - **Neural Monkey**, AllenNLP, Sockeye

- **Specialized applications**
  - Marian or tensor2tensor for NMT, Kaldi for ASR, etc.
Neural Monkey

- Open-source toolkit for sequential learning
- Suited for research and education
- Three (overlapping) user groups considered:
  - Students
  - Researchers
  - Newcomers to deep learning
Development

- Implemented in Python 3.6 using TensorFlow
- Thanks to TensorFlow GPU support using CUDA, cuDNN
- Actively developed using GitHub as the main communication platform

Source code here:

https://github.com/ufal/neuralmonkey
- Multimodal translation 
  (Charles University, ACL 2017, WMT 2018)
- Bandit learning 
  (Heidelberg University, ACL 2017)
- Graph Convolutional Encoders 
  (University of Amsterdam, EMNLP 2017)
- Non-autoregressive translation 
  (Charles University, EMNLP 2018)
Goals

1. Code readability
2. Modularity along research concepts
3. Up-to-date building blocks
4. Fast prototyping
Abstractions in Neural Monkey

- Compositional design
  - High-level abstractions derived from low-level ones
- (High-level) abstractions aligned with literature
  - Encoder, decoder, etc.
- Separation between model definition and usage
  - “Model parts” define the network architecture
  - “Graph executors” define what to compute in the TF session
**Usage**

- Neural Monkey experiments defined in INI configuration files
- Once config is ready, run with:
  
  `neuralmonkey-train config.ini`

- Inference from a trained model uses a second config for data:
  
  `neuralmonkey-run config.ini data.ini`
Simple MT Configuration Example

```
[main]
output="output_dir"
batch_size=64
epochs=20
train_dataset=<train_data>
val_dataset=<val_data>
trainer=<my_trainer>
runners=[<my_runner>]
evaluation=[
    ("target", evaluators.BLEU)]
logging_period=500
validation_period=5000

[en_vocabulary]
class=vocabulary.from_wordlist
path="en_vocab.tsv"

[de_vocabulary]
class=vocabulary.from_wordlist
path="de_vocab.tsv"

[train_data]
class=dataset.load
series=["source, target"]
data=[
    "data/train.en", "data/train.de"]

[val_data]
class=dataset.load
series=["source, target"]
data=[
    "data/val.en", "data/val.de"]

[my_encoder]
class=encoders.SentenceEncoder
rnn_size=500
embedding_size=600
data_id="source"

vocabulary=<en_vocabulary>

[my_attention]
class=attention.Attention
encoder=<my_encoder>
state_size=500

[my_decoder]
class=decoders.Decoder
encoders=[<my_encoder>]
attentions=[<my_attention>]
rnn_size=1000
embedding_size=600
data_id="target"
vocabulary=<de_vocabulary>

[my_trainer]
class=trainers.CrossEntropyTrainer
decoders=[<my_decoder>]
clip_norm=1.0

[my_runner]
class=runners.GreedyRunner
decoder=<my_decoder>
```
Configuration of Captioning

Define how to load images:

```
[imagenet_reader]
class=readers.image_reader.imagenet_reader
prefix="/mnt/troja/projects/wmt17-multimodal/data/flickr30k-images"
target_width=229
target_height=229
zero_one_normalization=True
```

And replace encoder with ImageNet network:

```
[imagenet_resnet]
class=encoders.ImageNet
name="imagenet"
data_id="images"
network_type="resnet_v2_50"
spatial_layer="resnet_v2_50/block4/unit_3/bottleneck_v2/conv3"
slim_models_path="tensorflow-models/research/slim"
```
Sentence Classification

Keep the encoder and replace the decoder (and update the rest)

```
[my_classifier]
class=decoders.Classifier
data_id="labels"
encoders=[<my_encoder>]
vocabulary=<label_vocabulary>
layers=[200]
```

```
[my_runner]
class=runners.PlainRunner
decoder=<my_classifier>
```

```
[my_trainer]
class=trainers.CrossEntropyTrainer
decoders=[<my_classifier>]
clip_norm=1.0
```

Pre-trained model parts
Parameters of model parts can be fixed using the gradient blocking module
Supported Features

- Recurrent encoder and decoder with attention
- Beam search decoding with model ensembling
- Deep convolutional encoder
- Self-attentive encoder and decoder (a.k.a. Transformer)
- Wrappers for ImageNet networks

- Custom CNNs for image processing
- ConvNets for sequence classification
- Self-attentive embeddings for sentence classification
- Hierarchical attention over multiple source sequences
- Generic sequence labeler
- Connectionist temporal classification
Console Logging during Training

Neural Monkey: A Natural Language Processing Toolkit
Scalar Values in TensorBoard

Losses, evaluation metrics, parameter norms, histograms of gradients
Attention in TensorBoard

TensorBoard: Scalars Images Graphs Distributions Histograms Projector Inactive

- Show actual image size

Runs

Write a regex to filter runs

- O san_ctc/
- O mn_mn/
- O san_ctc_proj/

Tags matching /.* (all tags)

- decoder/attention/image/0
  step 10,748,050  Wed Mar 14 2018 12:59:26 GMT+0100 (CET)
Neural Monkey is:

- Actively developing open-source GitHub project
- Suited for researchers, students, and other DL enthusiasts
- Collection of features from across the NLP sub-topics
- Simple to use because of clear and readable config files
- Highly modular, therefore relatively easy to debug

https://github.com/ufal/neuralmonkey