GTC 2019

Neural Monkey: A Natural Language Processing Toolkit

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

March 19, 2019

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

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)
  - TensorFlow, (py)Torch, Theano.
- 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 NLP tasks
- Suited for research and education
- Three (overlapping) user groups considered:
  - Students
  - Researchers
  - Newcomers to deep learning

Development

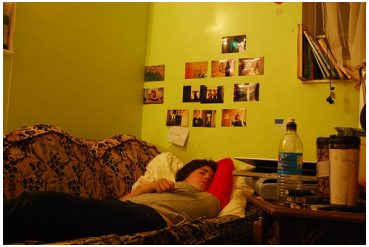
- Implemented in Python 3.6 using TensorFlow
- Thanks to TensorFlow GPU support using CUDA, cuDNN
- Actively developed using GitHub as the main communication platform

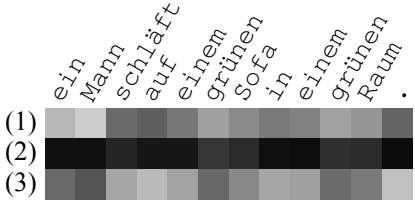
Source code here:

`https://github.com/ufal/neuralmonkey`

Used in Research

- 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

Goals

1. Code readability
2. Modularity along research concepts

Goals

1. Code readability
2. Modularity along research concepts
3. Up-to-date building blocks

Goals

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

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:

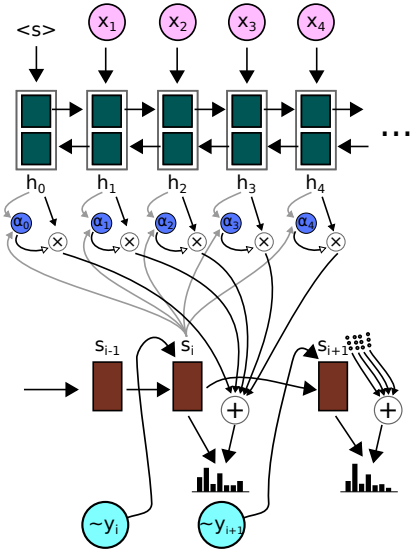
  ```
  neuralmonkey-run config.ini data.ini
  ```

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

Example Use-Case: Machine Translation

- Bahdanau et al., 2015
- Encoder: Bidirectional GRU
- Decoder: GRU with single-layer feed-forward attention



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>]
max_output_len=20
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>
output_series="target"
```

Simple MT Configuration Example

General training configuration:

```
[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
```

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", eva
logging_period=500
validation_period=5000

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

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

```
[train_data]
class=dataset.load
                    encoder=<my_encoder>
                    state_size=500
```

```
[my_decoder]
class=decoders.Decoder
                    _encoder>]
                    my_attention>]
                    n=20
                    e=600
                    et"
                    e_vocabulary>

                    s.CrossEntropyTrainer
                    _decoder>]

                    .GreedyRunner
                    decoder=<my_decoder>
                    output_series="target"
```

Loading vocabularies:

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

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

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", eva
logging_period=500
validation_period=5000

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

[de_vocabulary]
class=vocabulary.from_word
path="de_vocab.tsv"
```

Loading training and validation data:

```
[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"]
```

```
s.Decoder
_encoder>]
my_attention>]
n=20

e=600
et"
e_vocabulary>

s.CrossEntropyTrainer
_decoder>]

.GreedyRunner
ecoder>
="target"

state_size=500
```

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", eva
logging_period=500
validation_period=5000

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

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

```
[train_data]
class=dataset.load
```

```
class=attention.Attention
encoder=<my_encoder>
state_size=500
```

```
[my_decoder]
class=decoders.Decoder
_encoder>]
my_attention>]
n=20
e=600
et"
e_vocabulary>

s.CrossEntropyTrainer
_decoder>]

.GreedyRunner
decoder=<my_decoder>
output_series="target"
```

GRU Encoder configuration:

```
[my_encoder]
class=encoders.SentenceEncoder
rnn_size=500
embedding_size=600
data_id="source"
vocabulary=<en_vocabulary>
```

Simple MT Configuration Example

GRU Decoder and Attention configuration:

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

[my_decoder]
class=decoders.Decoder
encoders=[<my_encoder>]
attentions=[<my_attention>]
max_output_len=20
rnn_size=1000
embedding_size=600
data_id="target"
vocabulary=<de_vocabulary>
```

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", eva
logging_period=500
validation_period=5000

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

[de_vocabulary]
class=vocabulary.from_word
path="de_vocab.tsv"
```

Trainer and runner:

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

[my_runner]
class=runners.GreedyRunner
decoder=<my_decoder>
output_series="target"
```

```
s.Decoder
_encoder>]
my_attention>]
n=20

e=600
et"
e_vocabulary>

s.CrossEntropyTrainer
_decoder>]

.GreedyRunner
ecoder>
="target"

state_size=500
```

Configuration of Captioning

Define how to load images:

```
[imagenet_reader]
class=readers.image_reader.imagenet_reader
prefix="/home/test/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

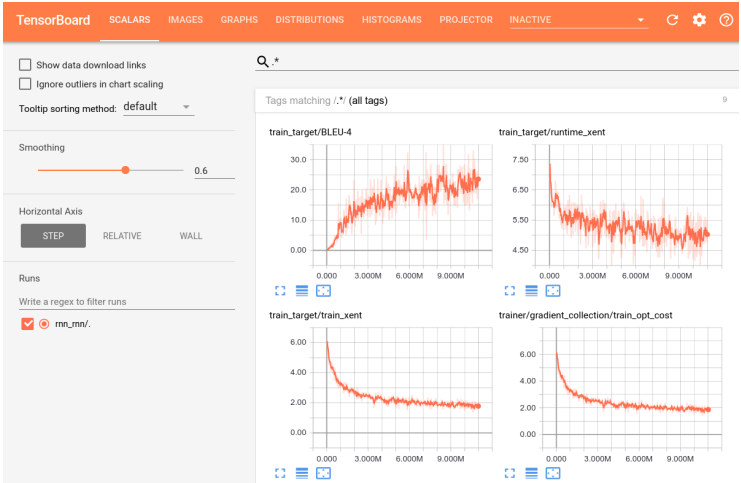
Supported Features

- 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

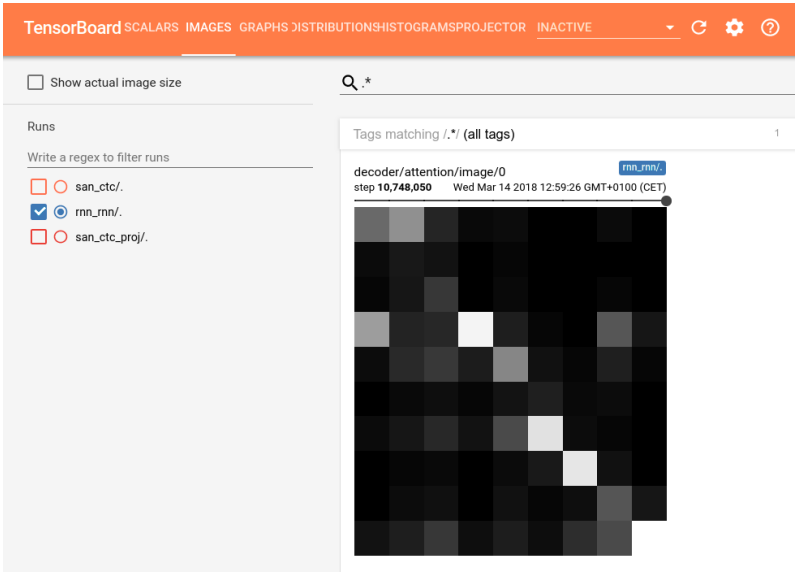
```
[11]
source: it started the French League season with two wins and two draws , and on Sunday it defeated St. Etienne 5 : 0 .
source_bpe: it started the French League season with two wins and two draws , and on Sunday it defeated St. Eti@@ enne 5 : 0 .
target_bpe: ten do se@@ z@@ ony francouzské ligy od@@ stat@@ oval dvěma výhra@@ mi a dvěma re@@ mi@@ zami , v neděli roz@@ střílel St . Eti@@ enne 5 : 0 .
target: začalo to <unk> <unk> <unk> <unk> a <unk> , a v neděli <unk> <unk> <unk> .
target (ref): ten do sezony francouzské ligy odstatoval dvěma výhrami a dvěma remízami , v neděli rozstřílel St . Etienne 5 : 0 .

[12]
source: it is putting the most pressure on AS Monaco .
source_bpe: it is putting the most pressure on AS Monaco .
target_bpe: relativně nejvíc tlačí bota klub AS Monaco .
target: <unk> <unk> tlak na <unk> AS .
target (ref): relativně nejvíc tlačí bota klub AS Monaco .

[13]
source: it so far has a balance of 1-1-2 in the league , and over the weekend it parted with Lille with a 1 : 1 tie .
source_bpe: it so far has a balance of 1-@@ 1-2 in the league , and over the weekend it parted with L@@ ille with a 1 : 1 tie .
target_bpe: ten má v lize zatím bilanci 1-@@ 1-2 a o víkendu se rozešel s L@@ ille s@@ mírně 1 : 1 .
target: v <unk> <unk> <unk> <unk> a <unk> se <unk> s <unk> <unk> <unk> .
target (ref): ten má v lize zatím bilanci 1-1-2 a o víkendu se rozešel s Lille smírně 1 : 1 .

[14]
source: today news appeared on tabloid English-language websites stating that the Queens Park Rangers may want Čech .
source_bpe: today news appeared on ta@@ blo@@ id Engli@@ sh-@@ language websites stating that the Queens Park Rangers may want Č@@ ech .
target_bpe: na bulvá@@ rních anglic@@ kých we@@ bech dnes vy@@ sko@@ čily zprávy , že by Č@@ ech@@ a mohli chtít v Queens Park Rangers .
target: dnes se objevila zpráva na internetových stránkách <unk> <unk> , které <unk> , že <unk> <unk> <unk> .
target (ref): na bulvárních anglických webech dnes vyskočily zprávy , že by Čecha mohli chtít v Queens Park Rangers .

[15]
source: however , according to English bookmakers , Čech &amp;apos;s departure from Chelsea at the last minute is not very likely .
source_bpe: however , according to English book@@ makers , Č@@ ech &@@ a@@ pos@@ ;@@ s departure from Chelsea at the last minute is not very likely .
target_bpe: podle anglic@@ kých book@@ ma@@ ker@@ Ů ale Č@@ ech@@ ův odchod z Chelsea na poslední chvíli není příliš pravděpodobný .
target: nicméně , podle <unk> <unk> , <unk> <unk> <unk> z Chelsea za poslední minutu není pravděpodobné .
target (ref): podle anglických bookmakerů ale Čechův odchod z Chelsea na poslední chvíli není příliš pravděpodobný .


2018-03-14 12:58:13: Validation (epoch 1, batch number 214960):
2018-03-14 12:58:46: Variable file saved in experiments/czeng_rnn_rnn/variables.data
2018-03-14 12:58:46: Best scores saved so far: [9.8540928005025079]
2018-03-14 12:59:06: Variables of 'encoder' saved to 'experiments/czeng_rnn_rnn/encoder.ckpt'
2018-03-14 12:59:26: Variables of 'encoder_input' saved to 'experiments/czeng_rnn_rnn/embeddings.ckpt'
2018-03-14 12:59:26: best target/BLEU-4 on validation: 9.855 (in epoch 1, after batch number 214960)
2018-03-14 12:59:26: Epoch 1/10  Instances 10748050  target/runtime_xent: 6.395    target/train_xent: 2.626    target/BLEU-4: 9.855
2018-03-14 12:59:26: Validation time: 11713.46s, inter-validation: 7391.83s, per-instance (train): 0.02s, per-instance (val): 4.41s
2018-03-14 12:59:26: Validation period setting is inefficient.

2018-03-14 13:03:11: Epoch 1/10  Instances 10748100  target/runtime_xent: 5.032    target/train_xent: 1.726    target/BLEU-4: 20.54
2018-03-14 13:15:41: Epoch 1/10  Instances 10778450  target/runtime_xent: 5.367    target/train_xent: 2.043    target/BLEU-4: 20.02
2018-03-14 13:28:07: Epoch 1/10  Instances 10808500  target/runtime_xent: 4.734    target/train_xent: 1.885    target/BLEU-4: 18.73
2018-03-14 13:40:35: Epoch 1/10  Instances 10838550  target/runtime_xent: 5.606    target/train_xent: 1.766    target/BLEU-4: 23.88
2018-03-14 13:53:03: Epoch 1/10  Instances 10868350  target/runtime_xent: 5.246    target/train_xent: 1.713    target/BLEU-4: 24.79
```

Scalar Values in TensorBoard



Losses, evaluation metrics, parameter norms, histograms of gradients

Attention in TensorBoard



Summary

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