NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning

Álvaro Peris Francisco Casacuberta
{lvapeab, fcn}@prhlt.upv.es

Pattern Recognition and Human Language Technologies
Research Center
Universitat Politècnica de València

Presentation Outline

Introduction 1

NMT-Keras 4

- Structure 4
- Interactive-predictive machine translation 11

Related projects 12

Future work 16

Introduction

- Keras: high-level deep learning API.
  - Works on top of Tensorflow, Theano or CNTK.
  - Layers as building blocks.
  - ☺ Code of quality and well organized.
  - ☺ Modular and extensible.
  - ☺ Good documentation, large community behind.

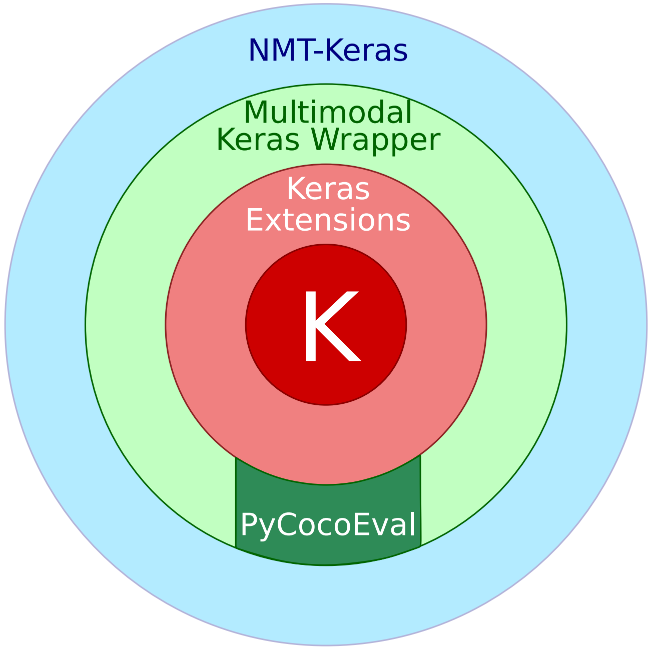
Introduction

- NMT-Keras: Neural machine translation with Keras.
  - Modular and extensible framework to NMT.
  - Easy usage of the library, but allowing the user to configure most of NMT options.
    - Tutorials and resources.
  - Several advanced/specific features:
    - Interactive-predictive NMT (INMT).
    - Continuous adaption of the models via online learning.
    - Active learning.

Introduction

- Integration via a wrapper: Multimodal Keras Wrapper.
  - Support for multimodal data.
  - Handles model loading/saving, data generators, data encoding, etc.
  - Trains and exploits the models.

NMT-Keras structure



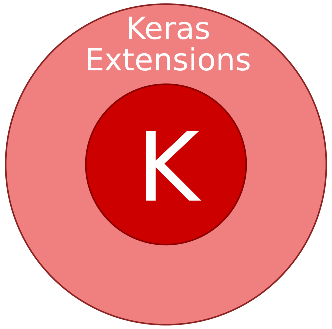
Keras

Deep learning framework.

- Layers:
  - RNNs (LSTM, GRU).
  - Embeddings.
  - CNNs.
  - FC layers.
  - ...
- Dropout, Batch Normalization, noise layers.
- Optimizers, initializers, regularizers.
- Parameter and activity regularizers and constraints.

Keras extensions

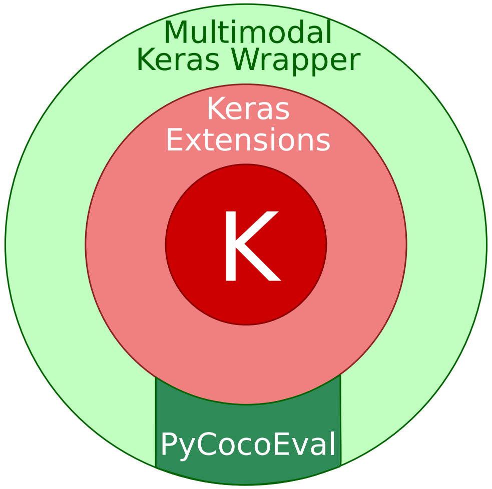
Keras' fork with extensions for sequence-to-sequence models (mainly).



- Updated with respect to the original repository.
- Attention mechanisms (add, dot, scaled-dot).
- RNNs with attention.
- Conditional GRU/LSTMs.
- Multi-head attention.
- Position-wise feed-forward.
- https://github.com/MarcBS/keras.

Multimodal wrapper

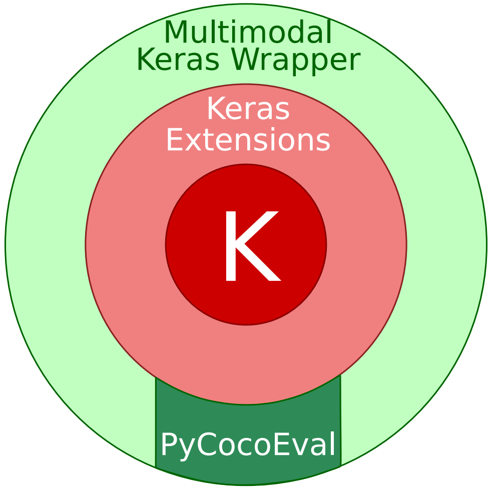
Eases the training and application of complex Keras models.



- Dataset object:
  - Manages the data: iterators, save/load.
  - Compatible with text, images, videos and categorical labels.
    - **Text:** vocabularies, shortlists, words ↔ indices.
    - **Image/video:** Pre-processing, data augmentation...

Multimodal wrapper

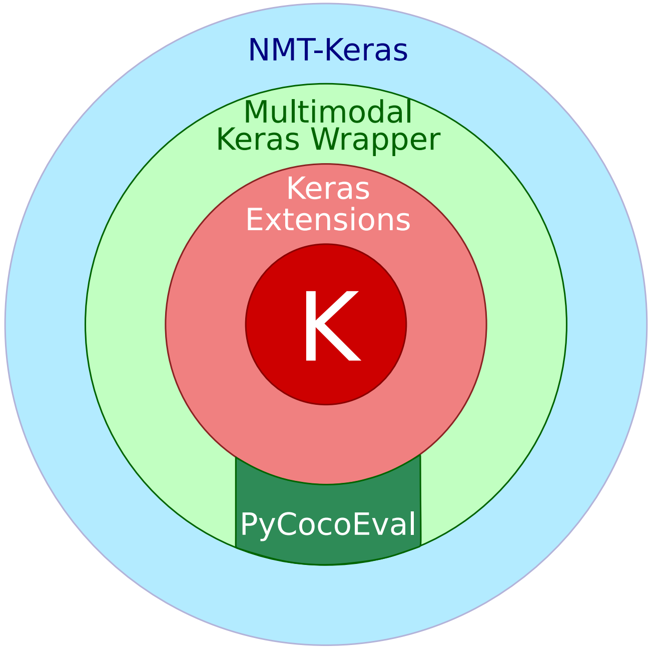
Eases the training and application of complex Keras models.



- `Model_Wrapper` object:
  - Manages the network logic:
    - Save/Load.
    - Training process.
    - Inference process.
  - Applies callbacks during training.
    - Periodical evaluation.
    - Early stop.
    - Learning rate schedules.
  - Beam search.
- Evaluation: `PyCocoEval` package.
  - BLEU, TER, METEOR, CIDEr and ROUGE-L.

NMT-Keras

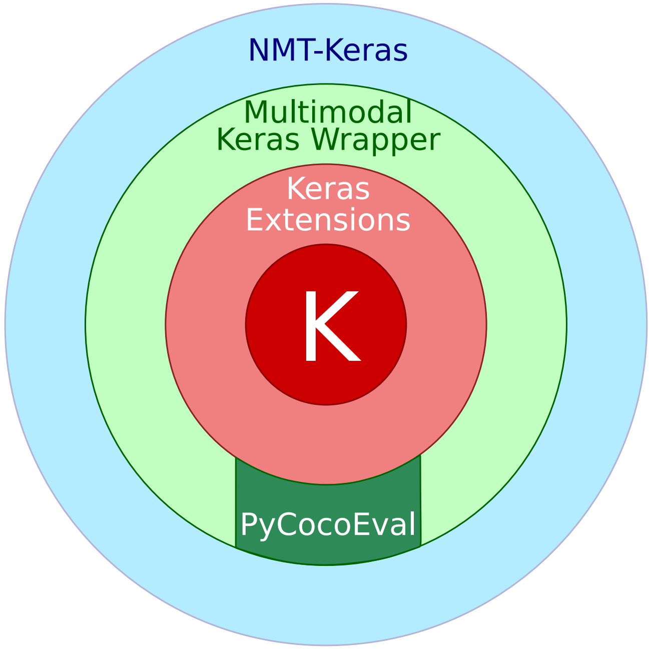
Toolkit for NMT based on Keras and Multimodal Keras Wrapper.



- Definition of models (model_zoo.py):
  - Deep attentional RNNs.
  - Transformer.
- Support for pre-trained embeddings.
- Ensemble decoding, N-best list generation, sentence scoring, model averaging, UNK replacement.
- Tutorials and examples available.
- Docs: `https://nmt-keras.readthedocs.io`.

NMT-Keras

NMT-Keras also features advanced features.



- Interactive-predictive neural machine translation.
- Online learning from post-edits or INMT.
- Active learning.
- Client-server architecture.

Interactive-predictive machine translation

- Efficient alternative to the regular post-editing of machine translation.

- Collaborative symbiosis between human and system.

  1. **User**: Introduces a correction to the system hypothesis.
  2. **System**: Provides an alternative hypothesis, considering the correction.

- Prefix-based corrections.

- Very suitable scenario for applying online learning.

Demo: http://casmacat.prhlt.upv.es/inmt/.

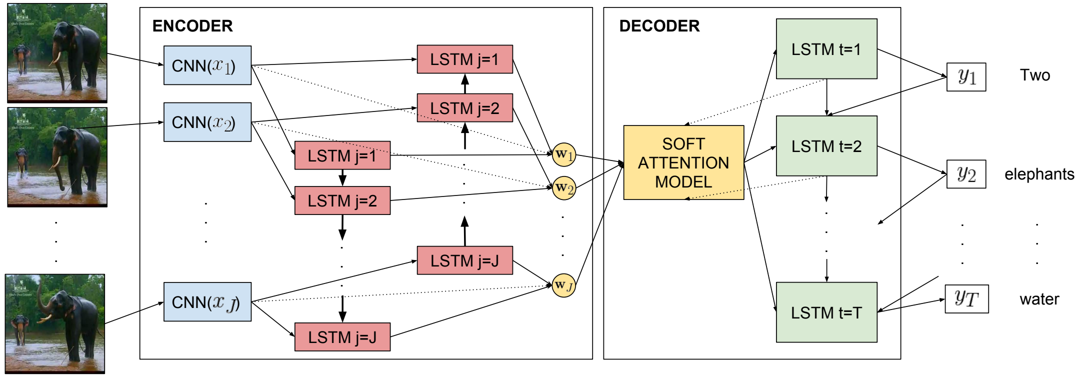
Source:		They are lost forever .
Target:		Ils sont perdus à jamais .
0	**MT**	Ils sont perdus pour toujours .
1	User MT	*Ils sont perdus* à pour toujours . *Ils sont perdus à* jamais .
2	**User**	*Ils sont perdus à jamais .*

Related projects

- Keras + Multimodal Wrapper:
  - High modularity.
  - Several problems can be addressed following this framework.
  - Handling of multimodal data.

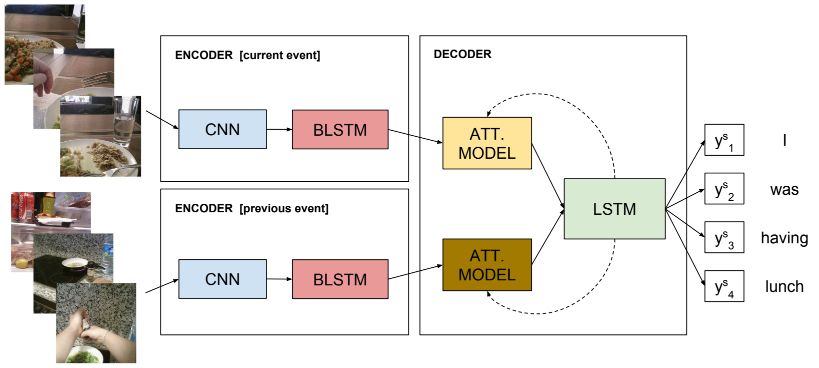
Related projects: Video captioning

Video Description using Bidirectional Recurrent Neural Networks.
Álvaro Peris, Marc Bolaños, Petia Radeva, Francisco Casacuberta.
@ICANN 2016.



https://github.com/lvapeab/ABiViRNet

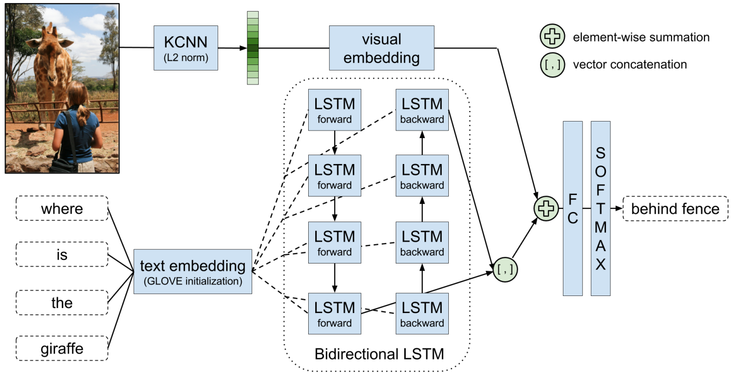
Egocentric video description based on temporally-linked sequences.
Marc Bolaños, Álvaro Peris, Francisco Casacuberta, Sergi Soler, Petia Radeva.
JVCIR 2018.



https://github.com/MarcBS/TMA

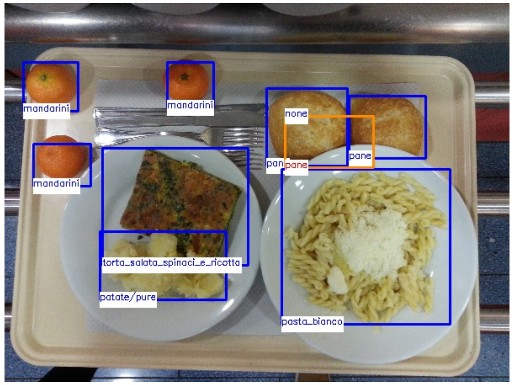
Related projects: Object classification

VIBIKNet: Visual Bidirectional Kernelized Network for Visual Question Answering.
Marc Bolaños, Álvaro Peris, Francisco Casacuberta, Petia Radeva.
VQA Challenge @CVPR 2016.



https://github.com/MarcBS/VIBIKNet

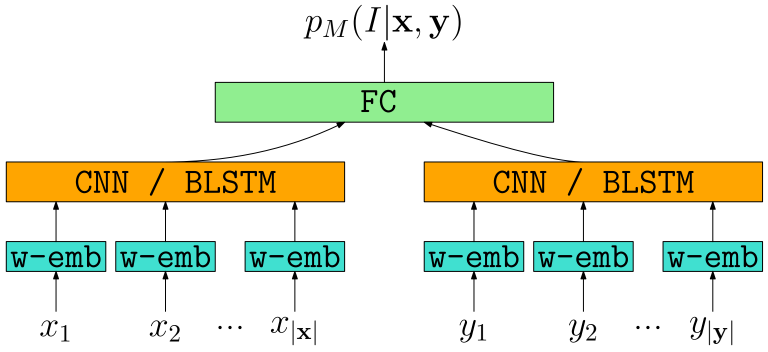
Grab, Pay and Eat: Semantic Food Detection for Smart Restaurants.
Eduardo Aguilar, Beatriz Remeseiro, Marc Bolaños, Petia Radeva.
arXiv:1711.05128. 2017.



Demo

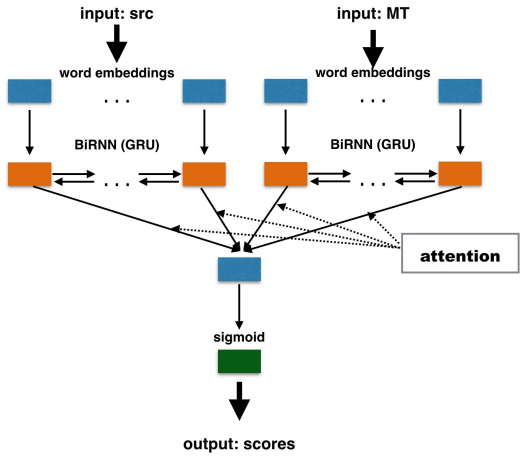
Related projects: Sentence classification

Neural Networks Classifier for Data Selection in Statistical Machine Translation.
Álvaro Peris, Mara Chinea-Rios, Francisco Casacuberta.
@EAMT 2017.



https://github.com/lvapeab/sentence-selectioNN

DeepQuest: a framework for neural-based Quality Estimation.
Julia Ive, Frédéric Blain, Lucia Specia.
@COLING 2018.



https://github.com/sheffieldnlp/deepQuest

Future work

- Extend features.
- Improve the website.
- Integrate parts of the Keras' fork into the main repository.

Contributions are welcome!!

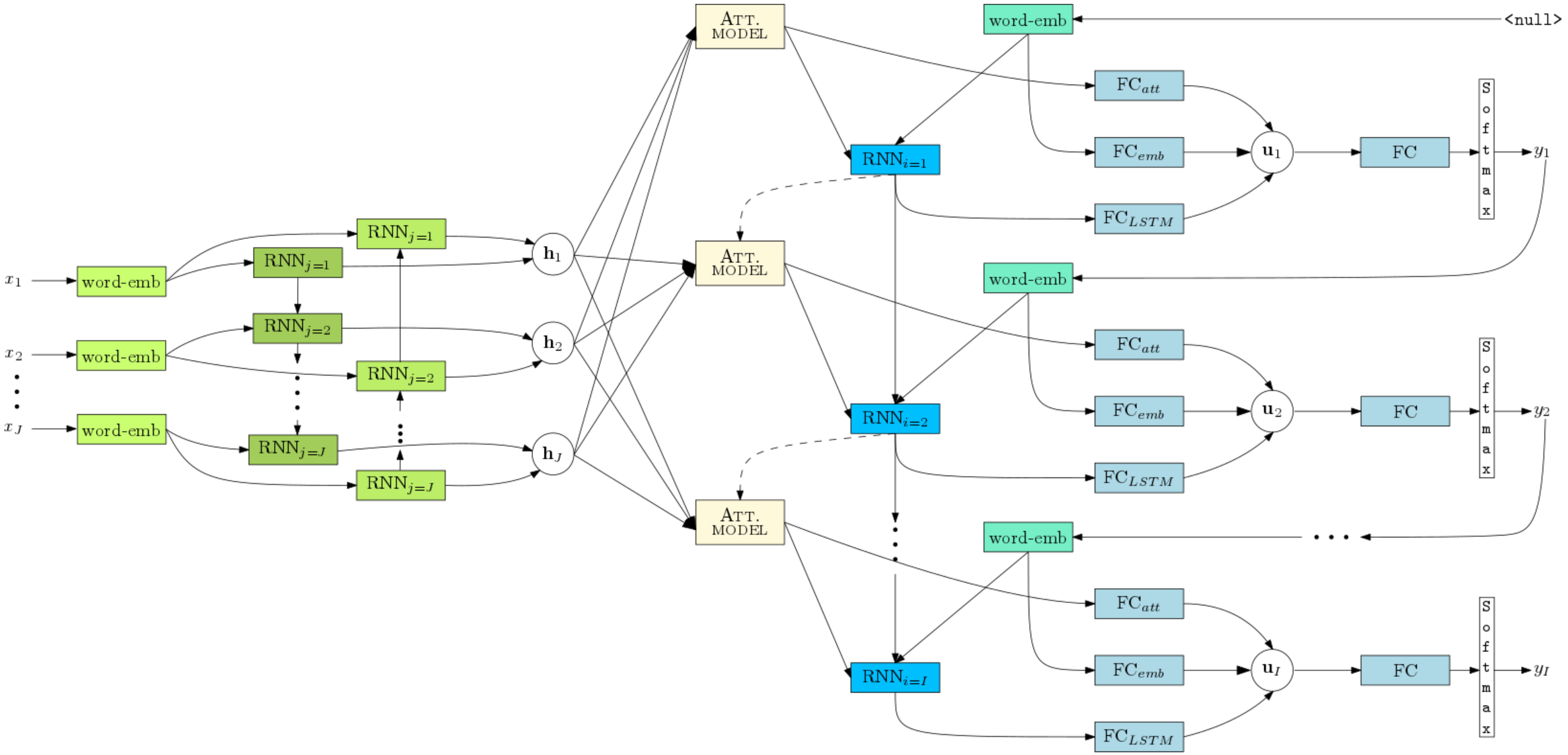
Thanks

Thank you!

Questions?

https://github.com/lvapeab/nmt-keras

Declaring a model in NMT-Keras



Declaring a model in NMT-Keras: Encoder

```
# model_zoo.py
# 1. Source text input
src_text = Input(name='source_text', batch_shape=tuple([None, None]), dtype='int32')
# 2. Encoder
# 2.1. Source word embedding
src_embedding = Embedding(input_vocabulary_size, embedding_size)(src_text)
# 2.2. Bidirectional encoder (GRU/LSTM)
annotations = Bidirectional(GRU(hidden_state_size, return_sequences=True))
(src_embedding)
```

Declaring a model in NMT-Keras: Decoder 1

```
# 3.10 Previously generated words as inputs for training
next_words = Input(name='state_below', batch_shape=tuple([None, None]),
  dtype='int32')
# 3.2. Target word embedding
state_below = Embedding(output_vocabulary_size, embedding_size)(next_words)
# 4.1. Initialize the decoder with a mean representation of the encoder.
ctx_mean = MaskedMean()(annotations)
initial_state = Dense(hidden_state_size, activation='tanh')(ctx_mean)
# 4.2. Decoder RNN
[proj_h, x_att, alphas, h_state] = AttGRUCond(
hidden_state_size,
return_sequences=True,
return_extra_variables=True)
([state_below, annotations, initial_state])
```

Declaring a model in NMT-Keras: Decoder 2

```
# 5.1 Add skip connections and deep output layers
out_layer_mlp = TimeDistributed(Dense(skip_size))(proj_h)
out_layer_ctx = TimeDistributed(Dense(skip_size))(x_att)
out_layer_emb = TimeDistributed(Dense(skip_size))(state_below)
# 5.1. Add and apply non-linearity
additional_output = Add()([out_layer_mlp, out_layer_ctx, out_layer_emb])
out_layer = Activation('tanh')(additional_output)
# 6. Softmax
out_probs = TimeDistributed(Dense(output_vocabulary_size, activation='softmax'),
name='target_text')(out_layer)
```

Instantiating and training the model

```
model = Model(inputs=[src_text, next_words], outputs=out_probs)
training_params = {'n_epochs': 100, 'batch_size': 40}
nmt_model.trainNet(dataset, training_params)
```

Basic usage

1. Set the desired configuration in config.py.

```
# [...]
MODEL_TYPE = 'AttentionRNNEncoderDecoder'
SOURCE_TEXT_EMBEDDING_SIZE = 512
TARGET_TEXT_EMBEDDING_SIZE = 512
N_LAYERS_ENCODER = 1
N_LAYERS_DECODER = 1
ENCODER_RNN_TYPE = 'LSTM'
DECODER_RNN_TYPE = 'ConditionalLSTM'
ENCODER_HIDDEN_SIZE = 512
DECODER_HIDDEN_SIZE = 512
# [...]
```

2. Launch `main.py` for training the model.

Basic usage

2.1 Build dataset

```
[16/12/2017 11:53:05] Running training.
[16/12/2017 11:53:05] Building ende dataset
[16/12/2017 11:53:16] Creating vocabulary for data with id 'target_text'.
[16/12/2017 11:53:41] Total: 27525 unique words in 1920209 sentences with a
  total of 54439548 words.
[...]
[16/12/2017 11:54:51] <<< Saving Dataset instance to Dataset_ende.pkl >>>
```

Basic usage

2.2 Build model

```
----------------------------------------------------------------------------------
Layer (type)                  Output Shape         Param #      Connected to
==================================================================================
source_text (InputLayer)     (None, None)                  0

----------------------------------------------------------------------------------
src_we (Embedding)           (None, None, 512)      9097216
  source_text[0][0]

----------------------------------------------------------------------------------
[...]

----------------------------------------------------------------------------------
tgt_text (TimeDistributed)   (None, None, 27528)   14121864      out_linear_0
==================================================================================
```

Basic usage

2.3. Compile and train

```
============================================================================
Total params: 49,931,145
Trainable params: 49,919,881
Non-trainable params: 11,264

--------------------------------------------------------------------------------
Preparing optimizer: Adam [LR: 0.0002 - LOSS: categorical_crossentropy - CLIP_C
  5.0 - CLIP_V 0.0 - LR_OPTIMIZER_DECAY 0.0] and compiling.
Epoch 1/5
1/38405 [..............................] - ETA: 11:12:22 - loss: 26.2976
[...]
36904/38405 [===========================>..] - ETA: 1:16:55 - loss: 6.8290
```

Basic usage

2.4. Finish training

```
[26/12/2017 01:01:26] <<< Model saved >>>
[26/12/2017 01:01:26] ---bad counter: 20/20
[26/12/2017 01:01:26] ---update 326250: early stopping.
Best Bleu_4 found at update 251250: 0.161165
```

Basic usage

4. Once the model has been trained, use it conveniently.

- Translate new text:

```
python sample_ensemble.py --text source.en --dest hyps.de
--dataset Dataset_ende.pkl --models NMT_Model_ende_update_251250
```

- Use model as scorer:

```
python score.py --text source.en --dest target.de --scores scores.ende
--dataset Dataset_ende.pkl --models NMT_Model_ende_update_251250
```

- Other applications.