Statistical Machine Translation: Neural Machine Translation

Jakub Waszczuk

Heinrich Heine Universität Düsseldorf

Winter Semester 2018/19

Outline

Outline

1. Neural Networks
2. RNN Encoder-Decoder
3. RNN Encoder-Decoder: Extentions

Neural networks

Abstraction

- Each neural network (NN) is simply a *function* with a set of *parameters*
- For instance from $\mathcal{R}^n$ to $\mathcal{R}^m$ for some $n, m > 0$

Construction

Construction of NNs is based on:

- Elementary building blocks (simple NNs: $\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots$)
- Combination operators (e.g. function composition $\mathbb{A} \circ \mathbb{B}$)
  $\rightarrow$ functional programming paradigm

However:

- We cannot use just any building block and any combination operation
- The building blocks should be *differentiable*
- The combination operators should preserve this property

Fortunately:

- Frameworks/domain specific languages make sure that we only build sane NNs

Network networks

Example

- **Code**: `https://github.com/kawu/bpfun` (look in the ‘src‘ directory)
- **Library**: backprop (`https://backprop.jle.im/index.html`)
  → automatic heterogeneous back-propagation library
  → allows to safely construct complex networks
- **Plan**: build a simple neural translation system

Feed-forward network

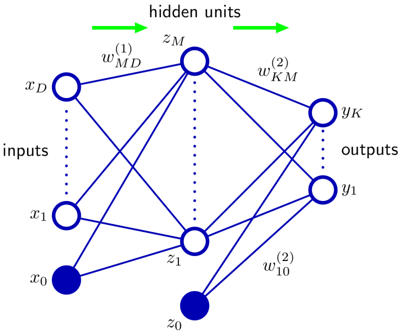
Network function:

$$y_k(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} z_j\right)$$

$$z_j = h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)$$

Graphical representation:

- Input, hidden, and output variables represented by nodes
- Weight parameters represented by links between nodes
- Arrows represent information flow during forward propagation
- (Source: [Bishop, 2006])



Expression power

Advantages:

- Neural networks are said to be *universal approximators*:

  "A two-layer [feed-forward] network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units." [Bishop, 2006]
- Seamless integration of word embeddings

Price to pay:

- Relatively complex model → lack of transparency
- Parameters hard to learn

Word embeddings

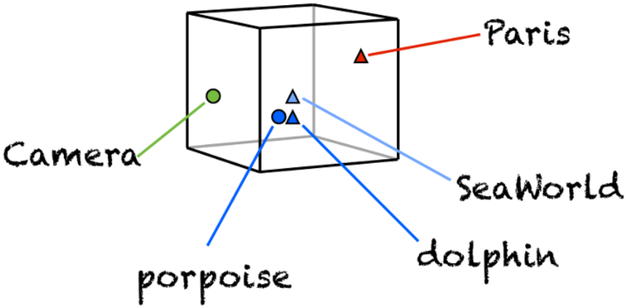


Figure: Representing words as N-dimensional vectors

Outline

Basic NMT System

Encoder-Decoder

- **Input**: concatenation of input word embeddings

$$v = x_1 \cdot x_2 \cdot \ldots \cdot x_n$$

- **Encoder**: calculate a vector representation of the output sentence

$$w = \mathbb{A}(v)$$

- **Decoder**: generate the output sentence from its hidden vector representation *w*

Basic NMT System

Encoder-Decoder

- **Input**: concatenation of input word embeddings

$$v = x_1 \cdot x_2 \cdot \ldots \cdot x_n$$

- **Encoder**: calculate a vector representation of the output sentence

$$w = \mathbb{A}(v)$$

- **Decoder**: generate the output sentence from its hidden vector representation w

Issues

- The size of the input vector v is not constant – it depends on the input length
- Feed-forward networks calculate vectors – how can we generate a sequence of words?

Recursive Neural Network

Recursive Neural Network (RNN)

- **Input**: sequence of vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$
- **Output**: sequence of vectors $\boldsymbol{h} = (h_1, \ldots, h_n)$
- **Recursive definition**:

$$\begin{aligned} h_i &= \mathbb{A}(x_i, h_{i-1}) \\ &= \mathbb{A}(x_i, \mathbb{A}(x_{i-1}, h_{i-2})) \\ &= \mathbb{A}(x_i, \mathbb{A}(x_{i-1}, \mathbb{A}(x_{i-2}, h_{i-3}))) \\ &= \mathbb{A}(x_i, \mathbb{A}(x_{i-1}, \mathbb{A}(x_{i-2}, \mathbb{A}(\ldots \mathbb{A}(x_1, h_0) \ldots)))) \end{aligned} \tag{1}$$

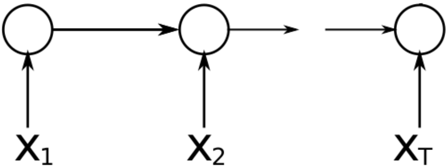
- **Parameters**: h_0 + those of $\mathbb{A}$

Intuitions

- h_i provides a summary of the prefix $x_1, \ldots, x_i$
- h_n provides a summary of the entire input sequence $\boldsymbol{x}$
  $\rightarrow$ **precisely what we need for encoding**

Recursive Neural Network

Graphical representation



Recursive Neural Network

Generative RNN

- **Input**: vector h_0 (here: representation of the the input sentence)
- **Hidden**: sequence of vectors $\boldsymbol{h} = (h_1, \ldots, h_m)$
- **Output**: sequence of words $\boldsymbol{y} = (y_1, \ldots, y_m)$ with $y_m = \text{EOS}$
- **Recursive definition**:

$$P(y_i \mid y_1, \ldots, y_{i-1}, \boldsymbol{x}) = \mathtt{softmax}(\mathbb{G}(h_{i-1})) \tag{2}$$

$$h_i = \mathbb{B}(y_i, h_{i-1}) \tag{3}$$

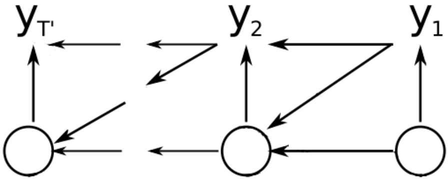
- **Parameters**: those of $\mathbb{B}$ and $\mathbb{G}$

Intuitions

- h_i provides a summary of the generated sentence $y_1, \ldots, y_i$, as well as information on what part of the input sentence has been already translated
- $\boldsymbol{y}$ represents the output sentence → **precisely what we need for decoding**

Recursive Neural Network

Graphical representation



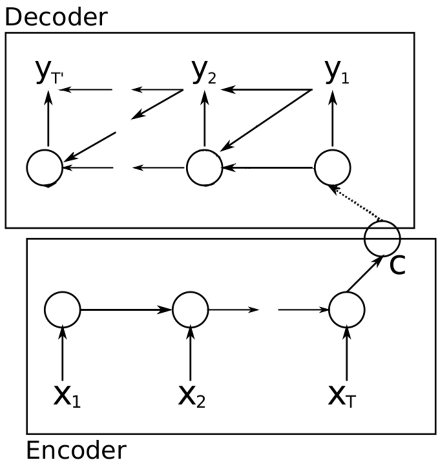
RNN Encoder-Decoder

Architecture

- **Input**: sequence of vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$
- **Encoding**: transform $\boldsymbol{x}$ into a **fixed-length** vector c
  $\rightarrow$ using standard RNN
- **Decoding**: generate a translation $\boldsymbol{y}$ from the encoded vector c
  $\rightarrow$ using „generative" RNN

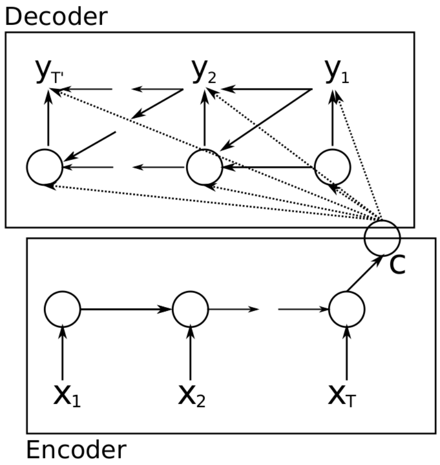
RNN Encoder-Decoder

Graphical representation



RNN Encoder-Decoder

Alternative architecture



RNN Encoder-Decoder

Training

- The two components of the RNN Encoder–Decoder are jointly trained to maximize the conditional likelihood of the training data:

$$\ell(D) = \prod_{(\boldsymbol{x},\boldsymbol{y}) \in D} P(\boldsymbol{y} \mid \boldsymbol{x}) \tag{4}$$

- Probability for a particular sentence pair is defined as:

$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{m} P(y_i \mid y_1, \ldots, y_{i-1}, \boldsymbol{x}) \tag{5}$$

where the individual $P(y_i \mid y_1, \ldots, y_{i-1}, \boldsymbol{x})$ are calculated by the network (see Eq. 2)

Encoder-Decoder in Practice

Phrase-based [Cho et al., 2014]

- Use the RNN Encoder-Decoder as a part of a standard phrase-based architecture
- Update the phrase translation probabilities using the RNN Encoder-Decoder
- Use the standard decoding algorithm on the modified phrase translation table

Seq2seq [Sutskever et al., 2014]

- The RNN Encoder-Decoder is directly used to translate
- The input sentence is encoded in a reversed direction (from end to start)

Outline

RNN Encoder-Decoder

Issues

RNN Encoder-Decoder

- is very simplistic
- works poorly for long sentences
- ignores 80% of what we have learned
- of course we can apply better optimization techniques, but is it all we can do?

Trends

One of the trends in deep learning for NLP:

- Benefit from what we know (about formal languages, linguistically inspired formalisms, classical computational models, etc.) in order to design better network architectures (see e.g. `https://sites.google.com/view/delfol-workshop-acl19`)

Possible Extensions

Encoder: ,,additive" semantics

- Assumption: the meaning of a sentence is the sum of the meanings of its words
  → that's obviously naive (and not true)
  → but a better prior assumption that nothing

Encoder: compositional semantics

- Assumption: the meaning of a sentence is a function of the meaning of its words and its syntactic (tree) structure
  → much more plausible
  → we could apply ,,tree2seq" network architectures
  → not easy because we typically don't know the structure of the input

Possible Extensions

Phrase-based-like decoding

Each time a new word is to be generated:

- Pick the relevant fragment (≈ phrase) in the input sentence
- Combine the vector representations of its words
- Based on that (and the previously generated words), generate the next word

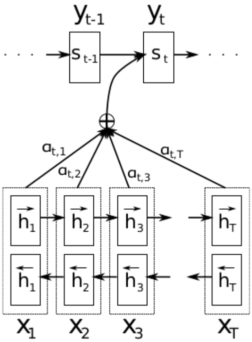


Figure: RNN Encoder-Decoder with Attention [Bahdanau et al., 2014]

RNN Encoder-Decoder with Attention

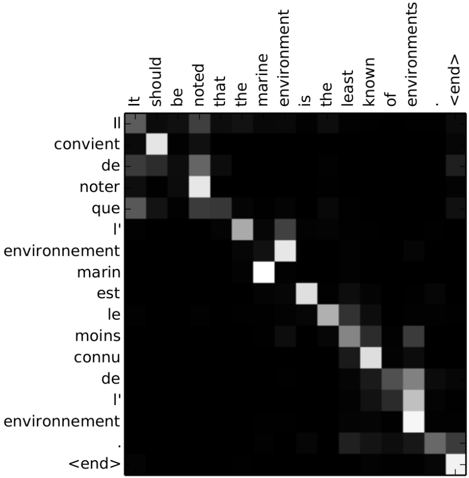


Figure: Example of alignments produced with attention-based RNN Encoder-Decoder

Discussion

Discussion

- Attention-based encoder-decoder can be seen as a variant of the phrase-based translation model couched in the framework of NNs
- It is one of the most influential NMT architectures nowadays
- Not necessarily SOTA, but attention is also employed in higher-scoring systems [Vaswani et al., 2017]

References I

Bahdanau, D., Cho, K., and Bengio, Y. (2014).
Neural machine translation by jointly learning to align and translate.
arXiv preprint arXiv:1409.0473.

Bishop, C. M. (2006).
Pattern Recognition and Machine Learning (Information Science and Statistics).
Springer-Verlag, Berlin, Heidelberg.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).
Learning phrase representations using rnn encoder–decoder for statistical machine translation.
In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
Sequence to Sequence Learning with Neural Networks.
In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017).
Attention is all you need.
In *Advances in Neural Information Processing Systems*, pages 5998–6008.