TECHNISCHE UNIVERSITÄT DRESDEN

SEMANTIC COMPUTING

Lecture 11: Deep Learning: Sequence to Sequence and Conclusion

Dagmar Gromann

International Center For Computational Logic

TU Dresden, 14 December 2018

Overview

- Sequence to Sequence
- Neural Machine Translation
- Practical Considerations

TECHNISCHE UNIVERSITÄT DRESDEN

Sequence to Sequence

Sequence to Sequence Models

Models that map an input sequence to an output sequence, an end-to-end approach.
Frequently, such models use encoder-decoder architectures, where they map the whole input sequence to one fixed-dimensional vector (using, e.g. an LSTM) and then decode the target sequence from that vector (with, e.g. another LSTM).

Great success in a variety of application tasks such as:

- neural machine translation
- speech recognition or generation
- text generation
- text summarization

Encoder-Decoder

Example of a standard encoder-decoder architecture, where the “encode" part represents one model (e.g. LSTM) and the “decode” part represents another model (e.g. CNN). The vector $[z_1, ..., z_d]$ is the output of the encoder and the input to the decoder.

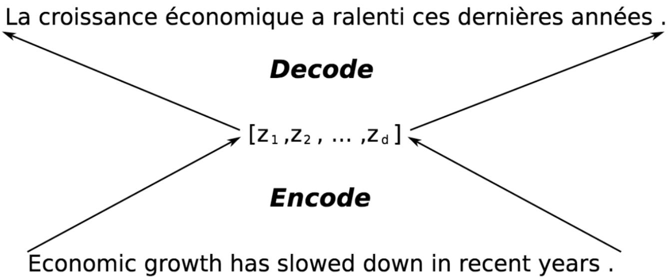
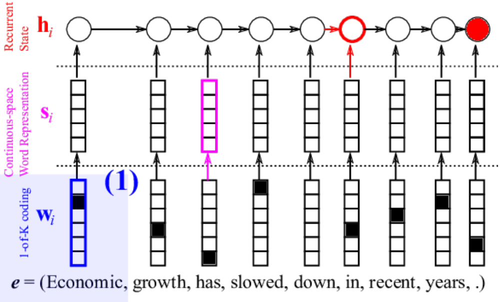


Image source: Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

Encoder

The encoder can be any architecture that produces an output vector, to be used as input by the decoder.



- w_i: one-hot vectors
- s_i: embedding matrix
- h_i: recurrent state (RNN, LSTM, GRU, etc.)

Output: one vector for the whole sequence (last red circle on the top right), i.e., a summary vector for the whole input sentence.

Image source: Cho, K. (2015). Introduction to Neural Machine Translation with GPUs. this is a link

Sentence Representation

The vector produced by the encoder corresponds to **sentence embeddings**.

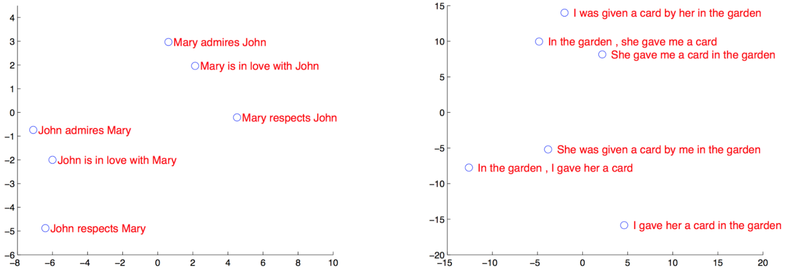
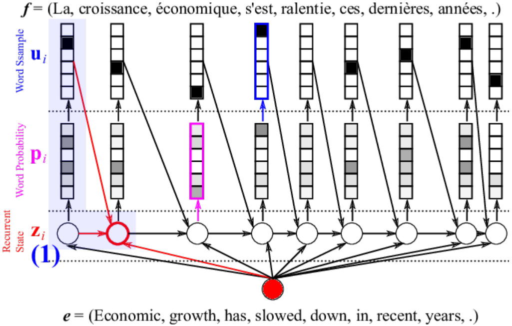


Image source: Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

Decoder

This example is specific to a type of RNN, could also be a CNN though.



- sentence representation vector h_T
- previous output word u_{i-1}
- previous state z_{i-1}

Image source: Cho, K. (2015). Introduction to Neural Machine Translation with GPUs. this is a link

Decoder Probabilities

The decoder probabilities:

- $e(k) = w_k^\top z_i + b_k$
- Probability of a word being the output word is computed by passing the previous formula through the softmax: $\frac{exp(e(k))}{\sum_j exp(e(j))}$
- once the output word u_i at time step i is selected, we iterate
  - update the hidden state z_{i+1}
  - compute probabilities for all target words p_{i+1}
  - predict output word u_{i+1}
- Iteration stops when predicting $< EOS >$ (end of sentence)
- In this example, the output sequence is longer than the input sequence

Training Encoder-Decoder

How can we train an encoder-decoder?

- large dataset of pairs of sentences to be translated
- Stochastic Gradient Descent (or any other optimizer)
- Backpropagation to compute the gradient

Neural Machine Translation

Finding the Optimal Output Sentence

Instead of a “greedy algorithm” that only takes the maximum probability of an output word, the **beam search** algorithm has been proposed as a more powerful choice to find the optimal output sentence.

- first step: determine the first k top words given their probabilities (k needs to be determined, e.g. 5)
- second step: extend conditional probability to $P(y^{(1)}|x)P(y^{(2)}|x, y^{(1)})$; memorize the 5 most likely words again (highest probability)
- third step: if the second step already excludes one of the options for the first word, this will not be considered in further steps
- iterate to $< EOS >$ ...

Attention

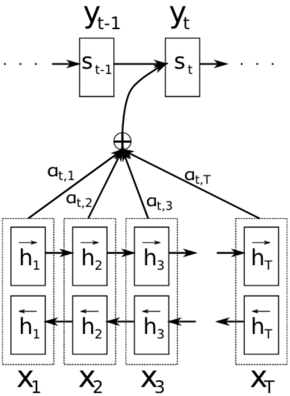
Attention is a mechanism to allow a model to automatically search for subparts of a source sequence as more relevant to predict a specific target word. To do this, it needs to have access to the last hidden state of the whole input sequence (retrieve from as needed).

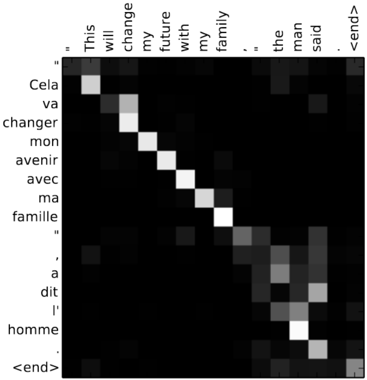
- Scoring: get a score for each hidden state of the input sequence (encoder) $score(h_i, ..., h_n)$
- Calculate softmax of score: $a_n(s) = \frac{e^s core(s)}{\sum e^s core(s')}$ which gives us a probability distribution of how much attention to pay to different parts of the source
- Then combine all hidden states of the encoder weighted by how much attention we pay to a context vector $c_t = \sum_s a_t(s) h_n$

Recent success: Attention is all you need

Bahdanau, D., Cho, K., and Bengio, Y. (2015) Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations.

Attention Continued



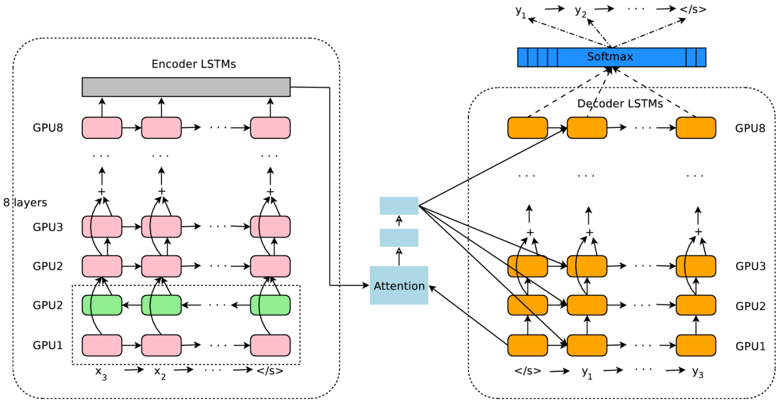


Source: Bahdanau, D., Cho, K., and Bengio, Y. (2015) Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations.

TECHNISCHE UNIVERSITÄT DRESDEN

Google's NMT model

Extended to multilingual translations.



Wu, Y. et al. (2016) Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

How to evaluate the model?

Human evaluation is expensive (time, cost) and language dependent.

- Current standard measure: **Bilingual Evaluation Understudy (BLEU)**.
- Idea: there are many possible good translations for a given sentence; compare n-grams of input sequence with n-grams of candidate translations (outputs)

Papineni, Kishore, et al. (2002). BLEU: a method for automatic evaluation of machine translation Proceedings of the 40th annual meeting on Association for Computational Linguistics.

BLEU score

Unigram precision only:

Candidate: the the the the the the the
Reference: the cat is on the mat

So $P = \frac{m}{w_t}$ where m is the number of candidate words that are in the reference (Z) and w_t is the total word count in the candidate, i.e., $P = \frac{7}{7}$

Modified n-gram precision where c_w is the number of times the word occurs in the reference and m_w is the number of times the word occurs in the candidate where the minimum of these two numbers is considered:

$$P = \frac{\sum_w min(c_w, m_w)}{w_t}$$

With a grain of salt...

Neural Machine Translation (NMT) systems should not be trusted blindly: In 2017 a man was arrested in Israel for posting “good morning” in Arabic with this picture, which was translated to “attack them” in Hebrew by Facebook’s NMT.

Image source: https://www.haaretz.com/israel-news/palestinian-arrested-over-mistranslated-good-morning-facebook-post-1.5459427

Practical Considerations

In machine learning

we always need to decide whether to

- gather more data
- increase/decrease model width/depth
- add/remove regularization
- improve optimization
- debug software implementation
- etc.

So knowing many different algorithms is not enough and we need to take principled decisions on the points above. This last section on Deep Learning seeks to provide some rules-of-thumb adapted from GoodFellow at al. (2016).

Guiding the Design Process

- error metrics and target values should be driven by the problem the application is intended to solve
- build an end-to-end system and use it to find bottlenecks (data, software, overfitting, underfitting, ...)
- incremental refinements (data-driven, hyperparameter-driven, ...)

Error Metrics

- Important decision that will influence the performance of your model
- Example: coverage = how many cases can the machine learning algorithm decide, how many cases have to be decided by humans; 100% accuracy if refusing to process any cases but reduces coverage to 0
- Other examples: click-through rates, satisfaction survey, etc.

Default Baseline

- build an end-to-end system
- copy state-of-the-art from related publication
- use this system as a baseline
- chance of being solved by linear weights? Logistic regression.
- AI-complete (speech recognition, machine translation, etc.)? Deep learning
  - fixed-sized vector input and supervised classification? Fully-Connected Feedforward NN
  - input and output sequence? LSTMs, GRUs, now CNNs also
  - images or time sequences? CNNs
- Use default optimization as a start, e.g. batch normalization and/or Adam
- Use default regularization, e.g. early stopping

General Baselines By Architecture

- Fully Connected Feedforward NN:
  - 2-3 hidden layers (Multi-Layer Perceptron (MLP))
  - ReLu, batch normalization, Adam, maybe Dropout
- Recurrent Neural Network (RNN):
  - LSTM, SGD, gradient clipping, high forget bias
- Convolutional Neural Network (CNN):
  - start out with pretrained network
  - OR: copy-paste architecture from related task + Batch normalization + Adam

Data-Driven Refinement

- high train error: inspect data (might be too noisy (!) - can a human process it?), tune hyperparameters, increase model depth/width
- high validation error: fine-tune hyperparameteris (manually, grid search, etc.)
- high test error: dataset augmentation, dropout, gathering more data

Review of Lecture 11

- What is Sequence to Sequence learning?
- What is an Encoder-Decoder model? Which NN architectures can it use?
- How does the output optimization for Neural Machine Translation work?
- How do we evaluate automatically produced translations?
- What is attention and what is it good for?
- Which considerations guide our design decisions and the design process of a deep learning model?