Lecture 7: Machine Translation, Sequence-to-Sequence and Attention

Presenter: Chu Đình Đức

Lecture Plan

1. Neural Machine Translation with Seq2Seq
2. Attention
3. Subword Models

1.1 Sequence-to-sequence Basics

- Seq2Seq model is an end-to-end model made up of two RNNs
  - a encoder, which takes the model's input sequence as input and encodes it into a fixed-size "context vector", and
  - a decoder, which uses the context vector from above as a "seed" from which to generate an output sequence
- Seq2Seq models are often referred to as "encoder-decoder models"

1.2 Seq2Seq architecture - encoder

- Because it's so difficult to compress an arbitrary-length sequence into a single fixed-size vector, the encoder will usually consist of stacked LSTMs
- Seq2Seq encoder will process the input sequence in reverse

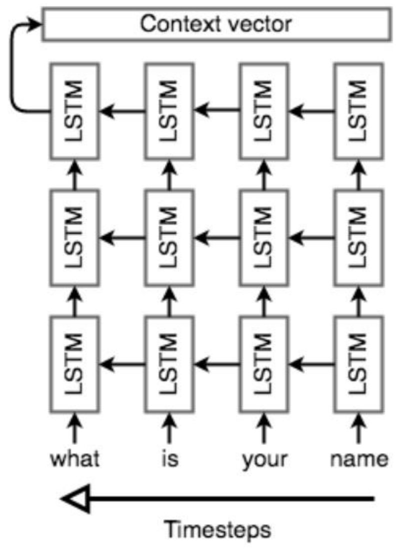

Figure 1: Example of a Seq2Seq encoder network. This model may be used to translate the English sentence "what is your name?" Note that the input tokens are read in reverse. Note that the network is unrolled; each column is a timestep and each row is a single layer, so that horizontal arrows correspond to hidden states and vertical arrows are LSTM inputs/outputs.

1.3 Seq2Seq architecture - decoder

- An <GO> token is appended to the end of the input (there's also one at the end of the output)
- Use softmax on the final layer's output to generate the output word, then pass that word into the first layer

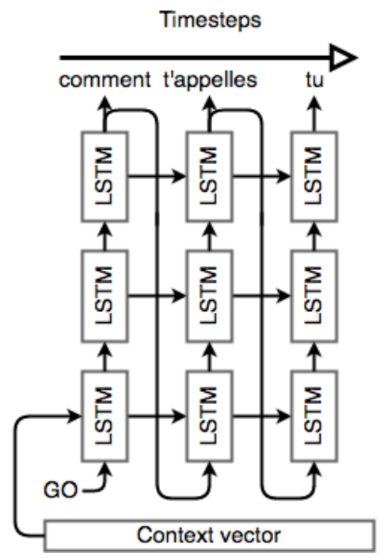

Figure 2: Example of a Seq2Seq decoder network. This decoder is decoding the context vector for "what is your name" (see Fig. 1 into its French translation, "comment t'appeles tu?" Note the special "GO" token used at the start of generation, and that generation is in the forward direction as opposed to the input which is read in reverse. Note also that the input and output do not need to be the same length.

1.4 Recap and Basic NMT Example

What a Seq2Seq model does in order to translate the English "what is your name" into the French "comment t'appelles tu"?

- We start with 4 one-hot vectors for the input, then embed them into 4 dense vector representation
- Encoder encodes 4 sequential vectors into a context vector
- Using this context vector and <GO> as input of decoder
- Decoder constructs "comment t'appelles tu" word by word (figure 2)

2.1 Motivation

- Different parts of a sentence have different levels of signification

"the ball is on the field"

- You primarily take note of the words "ball", "on" and "field"

2.2 Bahdanau et al. NMT model

- Input: x_1, …, x_n that we want to translate
- Output: y_1, …, y_m is target sentence
- Encoder:
  - Let (h1, …, hn) be hidden vectors representing the input sentence. These vectors are the output of a BiLSTM for instance, and capture contextual representation of each word in the sentence

2.2 Bahdanau et al. NMT model

- Decoder: we want to compute the hidden states s_i

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

    - s_{i-1} is the previous hidden vector
    - y_{i-1} is the generated word at the previous step
    - c_i is a context vector that capture the context from the original sentence that is relevant to the time step i of the decoder

2.2 Bahdanau et al. NMT model

- How to compute c_i?
  - For each hidden vector from the original sentence h_j, compute a score

$$e_{i,j} = a(s_{i-1}, h_j)$$

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{n} exp(e_{i,k})}$$

$$c_i = \sum_{j=1}^{n} \alpha_{i,j} h_j$$

  - Intuitively, this vector captures the relevant contextual information from original sentence for the i-th step of the decoder

2.3 Performance on long sentences

- The major advantage of attention-based models is their ability to efficiently translate long sentences
- As the size of the output grows, models that do not use attention will miss information and precision if they only use the final representation
- Attention is a clever way to fix this issue and experiments indeed confirm the intuition

3.1 Word segmentation

- Byte Pair Encoding
  - The essential idea is to start with a vocabulary of characters and keep extending the vocabulary with most frequent n-gram pairs in the dataset. Read more here
- After the vocabulary is built, an NMT system with some seq2seq architecture can be directly trained on these word segments.
- Notably, this method won top places in WMT 2016

3.2 Character-based model

- For each word w with m characters, instead of storing a word embedding, this model iterates over all characters $c_1, \ldots, c_m$ to look up the character embeddings $e_1, \ldots, e_m$
- These characters embeddings are then fed into a biLSTM to get the final hidden states h_f, h_b. The final word embedding is

$$e_w = W_f H_f + W_b H_b + b$$

3.3 Hybrid NMT

- Word-based Translation as a Backbone:
  - The core of the hybrid NMT is a deep LSTM encoder-decoder that translates at the word level
  - We maintain a vocabulary of size |V| per language and use <unk> to represent out of vocabulary words

3.3 Hybrid NMT

- The twofold advantage of such a hybrid approach is that:
  - It is much faster and easier to train than character-based ones
  - At the same time, it never produces unknown words as in the case of word-based models
- Word-based Translation as a Backbone
  - The core of hybrid NMT is a deep LSTM encoder-decoder that translates at the word level
  - We maintain a vocabulary of size |V| per language and use <unk> to represent out of vocabulary words

3.3 Hybrid NMT

- Source Character-based Representation
  - In regular word-based NMT, a universal embedding for <unk> is used to represent all out-of-vocabulary words
  - This is problematic because it discards valuable information about the source words
  - Instead, we learn a deep LSTM model over characters of rare words, and use the final hidden state of the LSTM as the representation for the rare word

3.3 Hybrid NMT

- Target Character-level Generation
  - General word-based NMT allows generation of <unk> in the target output. Instead, the goal here is to create a coherent framework that handles an unlimited output vocabulary
  - The solution is to have a separate deep LSTM that "translates" at the character level given the current word-level state. Note that the current word context is used to initialize the character-level encoder
  - The system is trained such that whenever the word-level NMT produces an <unk>, the character-level decoder is asked to recover the correct surface form of the unknown target word

3.3 Hybrid NMT

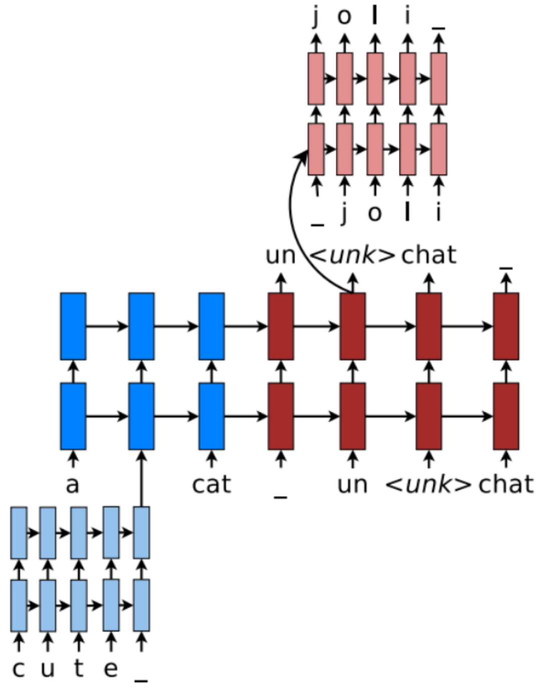

Figure 11: Hybrid NMT

THANK YOU!

Any questions