Lecture 8
Attention mechanism

LTAT.01.001 – Natural Language Processing
Kairit Sirts (kairit.sirts@ut.ee)
07.04.2021
Plan for today

● Sequence to sequence (encoder-decoder) models
  ● Using machine translation as an example
● Attention mechanism
● Different types of attention functions
● Applications of attention mechanism
Sequence to sequence (encoder-decoder) models
Recall sequence tagging models

- For each word predict one label
- The output sequence (labels) is directly aligned to the input sequence (words)
- Both sequences have the same length

The yellow cat is lazy
Sequence to sequence models

- Convert one sequence to another sequence

The red cat
Encoder-decoder models

- Convert one sequence to another sequence
Inside the seq2seq model

- Encoder computes the fixed length representation of the input
Inside the seq2seq model

- Encoder computes the fixed length representation of the input
- The encoded representation is passed as input to the decoder
- Decoder generates the output conditioned on the encoder representation and the previous output
Inside the seq2seq model

- Encoder computes the fixed length representation of the input
- The encoded representation is passed as input to the decoder
Inside the seq2seq model

- Encoder computes the fixed length representation of the input
- The encoded representation is passed as input to the decoder
- Generating stops after generating a dedicated stop symbol
Seq2seq model

- The encoder is typically a biLSTM
- The encoder representation is the concatenation of the final hidden states of the forward and backward LSTMs
- The decoder is a unidirectional LSTM
Seq2seq model

- Input to the next state of the decoder consists of the decoder last hidden state and the embedding of the previously predicted element
  
  \[ h_i^d = g(W^{(i)} y_{i-1} + W^{(h)} h_{i-1}^d) \]

- The initial hidden state of the decoder is set equal to the encoder representation where \( y_0 \) is a dedicated START symbol:
  
  \[ h_0^d = g(W^{(i)} y_0 + W^{(h)} h_n^e) \]

- The output at each step is predicted with a softmax:
  
  \[ p(y_j) = \text{softmax}(W^{(o)} h_i^d) \]
The problem with the simple seq2seq model formulation

- The input is representation is given to the model only in the beginning
- The later decoder states start quickly forgetting the input information
- Simple solution:
  - feed encoder representation at each decoder step
    \[ h_i^d = g(W^{(i)}y_{i-1} + W^{(h)}h_{i-1}^d + W^{(he)}h_n^e) \]
  - feed encoder representation also at each prediction step
    \[ p(y_j) = \text{softmax}(W^{(o)}h_i^d + W^{(oe)}h_n^e) \]
Seq2seq model with context info at every step
Training seq2seq models

- Similar to training a language model
- Minimize the cross-entropy of the predicted output wrt the gold output
- There are two options for training the decoder:
  - Feed the previous gold output as input
  - Feed the previous predicted output as input
Using a seq2seq model for generation

- Encode the input and start generating the output from the dedicated START symbol
- Several options for generating the next output:
  - Greedy: choose the output symbol with maximum probability
  - Sampling: sample the symbol from the output distribution
  - Beam search
- Continue generating until:
  - the dedicated STOP symbol is generated OR
  - the maximum output length is achieved
Beam search

- Beam search is a heuristic breadth-first search algorithm to find the approximate best solution (Viterbi was an exact algorithm)

- The main idea:
  - While decoding, store only a limited number of partial sequences that constitute a beam
  - Expand the sequences in the beam with all possible next steps and keep only the most probable.
  - Repeat, until the sequences are fully decoded.
Beam search with beam size 3

- Le 0.21
- La 0.15
- Un 0.06
- Une 0.02
- Je 0.001
Beam search with beam size 3

START

- Le 0.21
  - chiat 0.32
    - 0.0483
  - chat 0.23
    - 0.0075
  - girafe 0.05
    - 0.015
  - chiat 0.25
    - 0.009
- La 0.15
  - chat 0.15
    - 0.009
- Un 0.06
  - chiat 0.25
    - 0.009
- Une 0.02
  - chat 0.15
    - 0.009
- Je 0.001
Beam search with beam size 3
Beam search with beam size 3

START

Le 0.21

La 0.15

Un 0.06

Une 0.02

Je 0.001

0.0672

chat 0.32

0.0483

brun 0.17

0.0074

blanc 0.11

0.0075

girafe 0.05

0.0216

rouge 0.44

0.015

chat 0.25

0.0155

blanc 0.32

0.009

chat 0.15

0.0018

brun 0.12
Usages of seq2seq models

- Machine translation
- Chatbots/dialogues
- Image captioning
- Text summarization
- Speech synthesis
- Lemmatization
Problems with this model

- The whole input is crammed into a single fixed length vector
  - Perhaps it’s not a huge problem with short inputs but imagine machine translation of long sentences
- In many cases, the machine translation model has almost one to one relationships between inputs and outputs
  - the --> le
  - red --> rouge
  - cat --> chat
- In other cases, it must transform the input to output systematically
  - I speak → je parle
  - You speak → tu parles
- Simple seq2seq systems cannot learn these transformations effectively
Attention mechanism
What is attention?
Attention mechanism

- Attend to input while decoding

- Bahdanau et al., 2015. Neural machine translation by jointly learning to align and translate

- Encode each word in the input sentence with a bidirectional RNN

- When decoding, construct a conditioning context vector at each time step by computing a linear combination of the input vectors, weighted by the attention weights
Computing the attention

- "Query" – the decoder state
- “Keys” – the encoder states

- At each time step combine the “query” with each of the ”keys” to obtain scores indicating how important it is to attend to a particular input given the decoder state.
- Convert the scores into probabilities using softmax
- Then use these probabilities to compute the linear combination of the “key” vectors
Computing the attention

Attention weights:

\[ \alpha_1 = 0.01, \alpha_2 = 0.03, \alpha_3 = 0.74, \alpha_4 = 0.2, \alpha_5 = 0.01, \alpha_6 = 0.01 \]

Attention scores:

\[ a_1 = -1.2, a_2 = 0.1, a_3 = -3.4, a_4 = 2.1, a_5 = -1.0, a_6 = -0.5 \]

Key vectors from biLSTM:

Query:

Je parle

I speak with my red cat
Computing the attention

\[ c = \sum_i \alpha_i h_i \]

I speak with my red cat

Key vectors from biLSTM

Context vector

Attention weights

\[ \alpha_1 = 0.01 \quad \alpha_2 = 0.03 \quad \alpha_3 = 0.74 \quad \alpha_4 = 0.2 \quad \alpha_5 = 0.01 \quad \alpha_6 = 0.01 \]
Context

I * 0.01 + speak * 0.01 + with * 0.74 +

my * 0.2 + red * 0.01 + cat * 0.01
Decoder step with attention

- Compute attention weights $\alpha$ wrt to the current decoder state $h_{j-1}^d$
  - We’ll look in next slides how to do that
- Compute the context vector as a weighted sum of the encoder states
  \[ c = \sum_i \alpha_i h_i^e \]
- Compute the next decoder hidden state
  \[ h_j^d = g(y_{j-1}, h_{j-1}^d, c) \]
- Compute the next output
  \[ y_j = \text{softmax}(h_j^d, c) \]
Attention functions
Attention score functions

- \( q \) – query vector
- \( k \) – key vector

- **Multi-layer perceptron** (Bahdanau et al., 2015)
  \[
  a(q, k) = u^T \tanh(W[q; k])
  \]

- **Bilinear** (Luong et al., 2015)
  \[
  a(q, k) = q^T W k
  \]
Attention score functions

- **Dot product** (Luong et al., 2015)

\[ a(q, k) = q^T k \]

- No parameters. But the dimensionality of the key and query must be the same.

- **Scaled dot product** (Vaswani et al., 2017)

\[ a(q, k) = \frac{q^T k}{\sqrt{d_k}} \]
Computing attention weights

\[ a(q, k) = u^T \tanh(W[q; k]) \]

- **q** – query, the previous decoder state \( h_{j-1}^d \)
- **k** – keys, encoder states \( h_i^e \)

- With every encoder state \( h_i^e \), compute
  \[ a_i(h_{j-1}^d, h_i^e) = u^T \tanh(W[h_{j-1}^d; h_i^e]) \]
- Finally normalize with softmax
  \[ \alpha_i = \text{softmax}(\alpha_i) \]
Applications of attention mechanism

● The attention mechanism is used virtually everywhere in NLP
● Attention is an extremely useful module because it allows to pay attention to this part of the input that is relevant for the prediction
● It is also the basis for the Transformer architecture that underlie BERT, GPT and all these large language models.
● These models are the topic of the next week.
Gu et al. (2016). Incorporating Copying Mechanism in Sequence-to-Sequence Learning
Attend to different modalities

1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Attend to multiple sources (Huang et al., 2016)
Attention for text classification

- First encode words with attention into sentence representations.
- Then encode sentences with attention into document representation.
Using attention for interpretation

- Sentiment analysis: 4 – very positive, 0 – very negative

GT: 4 Prediction: 4

- pork belly = delicious.
- scallops?
- i don’t.
- even.
- like.
- scallops, and these were a-m-a-z-i-n-g.
- fun and tasty cocktails.
- next time i’m in phoenix, i will go back here.
- highly recommend.

GT: 0 Prediction: 0

- terrible value.
- ordered pasta entree.
- $16.95 good taste but size was an appetizer size.
- no salad, no bread no vegetable.
- this was.
- our and tasty cocktails.
- our second visit.
- i will not go back.
Self-attention for machine reading

Cheng et al. (2016). Long Short-Term Memory-Networks for Machine Reading