CS 886 Deep Learning and NLP

Ming Li
01. Word2Vec
02. Attention / Transformers
03. GPT / BERT
04. Simplicity, ALBERT, Single headed attention RNN
05. Student presentations Starting Feb. 3
06. Student presentations ending March 30
07. Student short presentations of research projects
Plan

1. Basic models, related to transduction models and attention.
2. Encoder-Decoder model, using recurrent networks such as LSTM.
3. Attention and Transformers
1. Fully connected network, feedforward network

To learn the weights on the edges
2. CNN

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that do convolutional operation.
Each filter detects a small pattern (3 x 3). These are the network parameters to be learned.

**Convolutional layer**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

**Filter 1**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Filter 2**

<table>
<thead>
<tr>
<th></th>
<th>-1</th>
<th>1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>
Convolution Operation

**Input**

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

**Filter 1**

\[
\begin{array}{ccc}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1 \\
\end{array}
\]

**stride=1**

-1-1

3 -1

Dot product
Convolution

Input

stride=1

Filter 1
3. RNN

Parameters to be learned: U, V, W
Simple RNN vs LSTM

(a) RNN

(b) LSTM

Attention and Transformers
Attention and Transformers

Encoder-Decoder machine translation

\[
f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .})
\]

\[
e = (\text{Economic, growth, has, slowed, down, in, recent, years, .})
\]
Encoder-Decoder LSTM structure for chatting (for non-intelligent beings)
Attention and Transformers

Attention

\[ f = (La, \text{ croissance, \text{ économique, s'est, ralentie, ces, dernières, années, .})} \]

\[ e = (Economic, \text{ growth, has, slowed, down, in, recent, years, .}) \]
A vector for each region

$z^0$ is initial parameter, it is also learned

0.7

Image caption generation using attention
Image caption generation using attention

A vector for each region

Word 1

Attention to a region

weighted sum

0.7 0.1 0.1 0.0 0.0
Attention and Transformers

Image caption generation using attention

A vector for each region

Word 1

Word 2

weighted sum

0.0 0.8 0.2

0.0 0.0 0.0 0.0
Image caption generation using attention

A woman is throwing a frisbee in a park. A dog is standing on a hardwood floor. A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear. A group of people sitting on a boat in the water. A giraffe standing in a forest with trees in the background.

Many new ideas

1. ULM-FiT, pre-training, transfer learning in NLP
2. Recurrent models require linear sequential computation, hard to parallelize. ELMo, bidirectional LSTM.
3. In order to reduce such sequential computation, several models based on CNN are introduced, such as ConvS2S and ByteNet. Dependency for ConvS2S needs linear depth, and ByteNet logarithmic.
4. The transformer is the first transduction model relying entirely on self-attention to compute the representations of its input and output without using RNN or CNN.
Attention and Transformers

Transformer

INPUT: Je suis étudiant

OUTPUT: I am a student
An Encoder Block: same structure, different parameters
Attention and Transformers

Note: The ffnn is independent for each word. Hence can be parallelized.
First we create three vectors by multiplying input embedding (1x512) $x_i$ with three matrices (64x512):

$q_i = x_i W^Q$

$K_i = x_i W^K$

$V_i = x_i W^V$
Self Attention

Now we need to calculate a score to determine how much focus to place on other parts of the input.
Self Attention

Formula

$$d_k = 64$$ is dimension of key vector
Multiple heads

1. It expands the model’s ability to focus on different positions.
2. It gives the attention layer multiple “representation subspaces”
1) Concatenate all the attention heads

2) Multiply with a weight matrix $W^0$ that was trained jointly with the model

The output is expecting only a $2 \times 4$ ($|\text{input}| \times 64$) matrix, hence,

3) The result would be the $Z$ matrix that captures information from all the attention heads. We can send this forward to the FFNN

$Z = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$
Representing the input order (positional encoding)

The transformer adds a vector to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word, or the distance between different words in the sequence. The intuition here is that adding these values to the embeddings provides meaningful distances between the embedding vectors once they’re projected into Q/K/V vectors and during dot-product attention.

Can somebody present positional encoding following https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
Add and Normalize

In order to regulate the computation, this is a normalization layer so that each feature (column) have the same average and deviation.
Layer Normalization (Hinton)

Layer normalization normalizes the inputs across the features.
The encoder-decoder attention is just like self attention, except it uses K, V from the top of encoder output, and its own Q.
Decoder's Output

Get the index of the cell with the highest value \( \text{argmax} \)

**log_probs**

| 0 | 1 | 2 | 3 | 4 | 5 | ... vocab_size |

**logits**

| 0 | 1 | 2 | 3 | 4 | 5 | ... vocab_size |

Decoder stack output

**Softmax**

**Linear**

Which word in our vocabulary is associated with this index?

am
Attention and Transformers

How it works

But what about self-attention?
But what about self-attention when the input is “incomplete”?

The solution is to set future unknown values with “-inf”.

The same for Encoder-Decoder Attention.
We can use cross Entropy.

We can also optimize two words at a time: using BEAM search: keep a few alternatives for the first word.
Cross Entropy and KL (Kullback-Leibler) divergence

• **Entropy**: \( E(P) = - \sum_i P(i) \log P(i) \) - expected code length (also optimal)
• **Cross Entropy**: \( C(P) = - \sum_i P(i) \log Q(i) \) – expected coding length using optimal code for \( Q \)
• **KL divergence**: 
  \[
  D_{KL}(P \parallel Q) = \sum_i P(i) \log[P(i)/Q(i)] = \sum_i P(i)[\log P(i) - \log Q(i)], \text{ extra bits}
  \]
• **JSD** (\( P \parallel Q \)) = \( \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M) \), \( M = \frac{1}{2} (P+Q) \), symmetric KL

* JSD = Jensen-Shannon Divergency
# Transformer Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td>ByteNet [18]</td>
<td>23.75</td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk [39]</td>
<td>24.6</td>
<td>40.4</td>
</tr>
<tr>
<td>GNMT + RL [38]</td>
<td>25.16</td>
<td>40.46</td>
</tr>
<tr>
<td>ConvS2S [9]</td>
<td>26.03</td>
<td>40.56</td>
</tr>
<tr>
<td>MoE [32]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk Ensemble [39]</td>
<td></td>
<td>40.4</td>
</tr>
<tr>
<td>GNMT + RL Ensemble [38]</td>
<td>26.30</td>
<td>41.16</td>
</tr>
<tr>
<td>ConvS2S Ensemble [9]</td>
<td>26.36</td>
<td>41.29</td>
</tr>
<tr>
<td>Transformer (base model)</td>
<td>27.3</td>
<td>38.1</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>28.4</td>
<td>41.8</td>
</tr>
</tbody>
</table>
Attention and Transformers

Next Lecture

BERT

GPT
Vaswani et al. Attention is all you need. 2017.

Resources:

https://nlp.seas.harvard.edu/2018/04/03/attention.html (Excellent explanation of transformer model with codes.)

Jay Alammar, The illustrated transformer (from which I borrowed many pictures):

http://jalammar.github.io/illustrated-transformer/

Kate Logninova: Attention in NLP, summarizes all sorts of attentions. Can somebody present this and related literature? https://medium.com/@joealato/attention-in-nlp-734c6fa9d983