Vision & Language

RNNs, Captioning, Attention, Transformers
First Assignment

• Due very soon...!
Course Project

• Chat group on Campuswire – **Introduce yourselves** if nothing more...

• Start forming your group and start working on a one-page to two-page project proposal

• Groups from 1 to 3 students. (You can work on your own)

• Project effort should be equivalent to at least one of the assignments – keep in mind this semester ends a bit short – so think of your project as your Assignment #4 (for grad students), Assignment #3 (for undergrad students).

• So Project should be like an Assignment #4 – but it is yours. I won’t push you to do anything but it should hopefully be relevant to the class topic – vision and language. e.g. not prediction of the weather using ML – or email spam classification.

• **Project Group Formation Enabled – Project Proposal due Soon!**
Last Class

• Recap on Word Embeddings (CBOW)
• More on Tokenization
• Recurrent Neural Network – Transition Cell
• Recurrent Neural Network (Unrolled)
• Understanding Issues with Batching...
• Variations with gated connections: LSTMs and GRUs
• Stacked and Bidirectional RNNs
• Use in vision and language: Neural Image Captioning
Today

- Sequence-to-sequence (RNNs) for Machine Translation
- Learning to Align and Translate with Soft Attention
- Image Captioning (CNNs + RNNs): Show and Tell
- Image Captioning (CNNs + RNNs + Attention): Show Attend and Tell
- Attention is All you Need!
- Encoder Transformers: BERT
- Decoder Transformers: GPT-2 – maybe next class
RNNs – One-to-one sequence prediction
RNNs – Sequence to score prediction

Classify

[English, German, Swiss German, Gaelic, Dutch, Afrikaans, Luxembourgish, Limburgish, other]
RNNs for Text Generation (Auto-regressive)
RNNs for Machine Translation Seq-to-Seq

The world is not enough.

El mundo no es suficiente.
Perhaps a better idea is to compute the average $h$ vector across all steps and pass this to the decoder.
Perhaps an even better idea is to compute the average $h$ vector across all steps and pass this to the decoder at each time step in the decoder!

$$\bar{h} = \frac{1}{n} \sum h_i$$
Perhaps an even better idea is to compute the average $h$ vector across all steps and pass this to the decoder at each time step in the decoder **but using a weighted average with learned weights!!**
Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder at each time step in the decoder but using a weighted average with learned weights, and the weights are specific for each time step!!!

\[
\bar{h}_j = \sum a_{j,i} h_i
\]

such that:

\[
a_{j,i} = \frac{\exp(h_j v_{j-1})}{\sum \exp(h_i v_{i-1})}
\]
Let’s take a look at one of the first papers introducing this idea.

**Figure 1:** The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$.

\[
c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.
\]

\[
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},
\]

\[
e_{ij} = a(s_{i-1}, h_j)
\]
Let’s look at the Attention weights
CNNs + RNNs for Image Captioning

Vinyals et al. Show and Tell: A Neural Image Caption Generator
https://arxiv.org/abs/1411.4555
References (a lot of them)


CNNs + RNNs for Image Captioning w/ Attention

Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention


Only showing the third time step encoder-decoder connection

convert to 49 vectors of size 512 and those become \( h_i \)

output tensor of size \( C \times H \times W \) e.g. 512x7x7
Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

Fixed number of input tokens
[but hey! we can always define a large enough length and add mask tokens]
Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

Fixed number of input tokens
[but hey! we can always define a large enough length and add mask tokens]
Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

Fixed number of input tokens
[but hey! we can always define a large enough length and add mask tokens]

Encoder

Decoder

Multi-head Self Attention Module (Transformer)
We can also draw this as in the paper:

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762
Regular Attention: + Scaling factor

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
\]
This is not unlike what we already used before

V: those are h’s here
Q: those are h’s here
K: those are v’s here

Only showing the third time step encoder-decoder connection

\[ \tilde{h}_j = \sum a_{j,i} h_i \]

such that:

\[ a_{j,i} = \frac{\exp(h_jv_{j-1})}{\sum \exp(h_i v_{i-1})} \]
Multi-head Attention: Do not settle for just one set of attention weights.

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O
\]

where \( \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \)

Where the projections are parameter matrices \( W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k} \), \( W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k} \), \( W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v} \), and \( W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}} \).
We can lose track of position since we are aggregating across all locations

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$
Multi-headed attention weights are harder to interpret obviously.
The BERT Encoder Model


**Important things to know**

- No decoder

- Train the model to fill-in-the-blank by masking some of the input tokens and trying to recover the full sentence.

- The input is not one sentence but two sentences separated by a [SEP] token.

- Also try to predict whether these two input sentences are consecutive or not.
The BERT Encoder Model

UNITER: UNiversal Image-TExt
Representation Learning

Yen-Chun Chen*, Linjie Li*, Licheng Yu*, Ahmed El Kholy
Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu

Microsoft Dynamics 365 AI Research
{yen-chun.chen,lindsey.li,licheng.yu,ahmed.elkholy,fiahmed,
zhe.gan,yu.cheng,jingjl}@microsoft.com
Questions?