Attention

Vojtěch Jindra

Outline

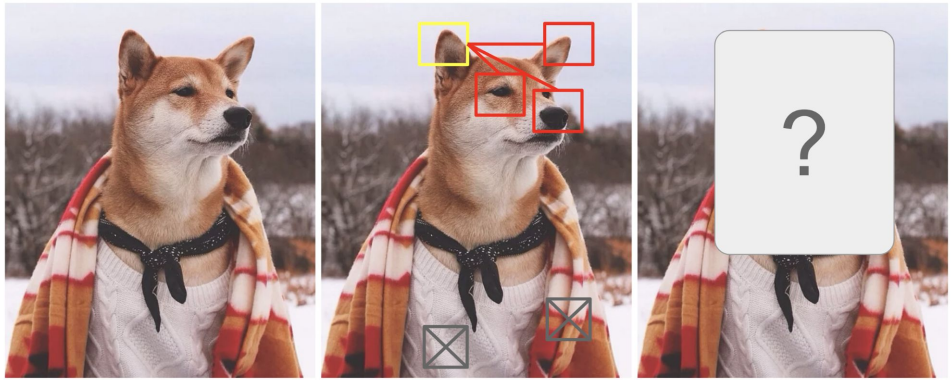
1) Motivation for introducing attention
2) The intuition behind attention
3) Transformer
4) Set Transformer
5) Variations of attention
6) Applications
7) Demo
8) Further reading
9) References

Motivation

- Initially used to improve seq2seq[1] models
- The idea of attention has spread to other applications as well
  - Neural Machine Translation (seq2seq)[2]
  - Speech recognition[3]
  - Graph attention networks[4]
  - Recommender systems[5]
  - Self-driving cars[6]
- Seq2seq models generate outputs iteratively from start of the sequence to its end
- With long sequences, the model tends to put less attention towards the start of the sequence
- Attention calculates a similarity vector between words to tell the model when to focus on what words

The intuition behind attention

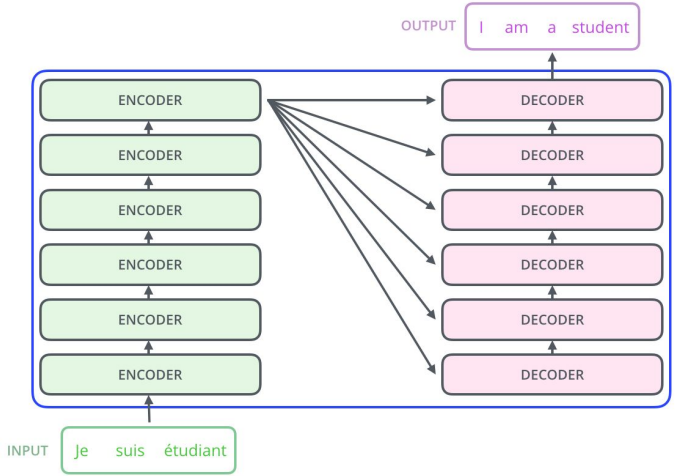
- Humans tend to put more attention towards words they want to translate in a sentence
  - When translating the sentence “Dude, look here!” to spanish (“¡Tío, mira aquí!”), one will put more attention towards the relationship between “Dude” and “Tío” than “Dude” and “aquí”
- Similarly, when figuring out what a picture is about, we put more more attention towards specific treats
  - When trying to classify if an animal is a dog, we will put more attention towards the dog-like features (e.g. ears) rather than to the general animal-like features (e.g. eyes)

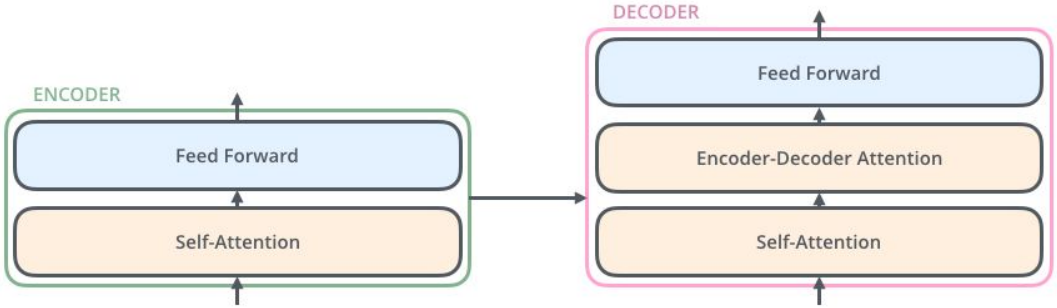


https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Transformer

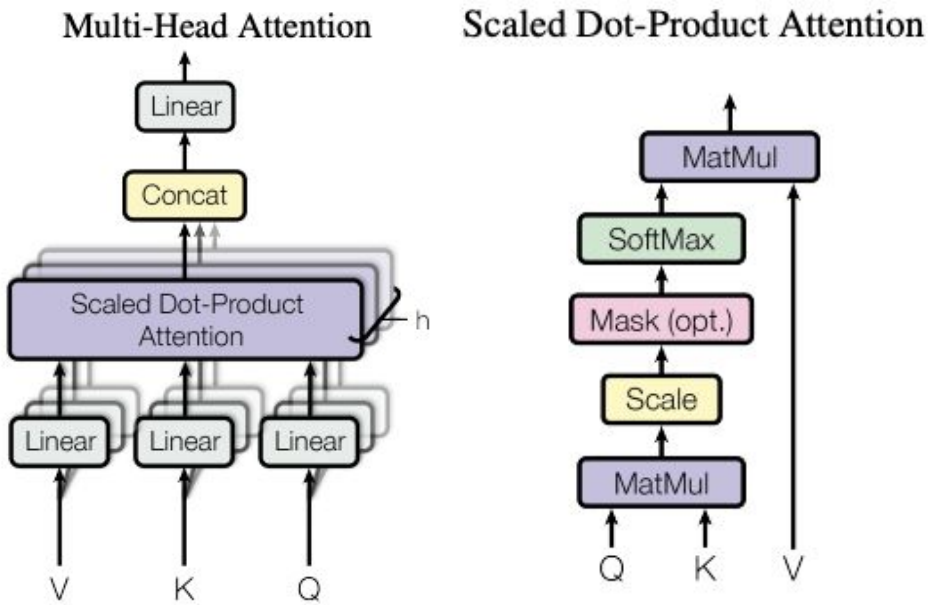
- The seq2seq with attention blackbox is called a Transformer
- Transformer is composed out of a stack of encoders and decoders (originally six of each)
- Inputs go sequentially through the encoders and then from the last encoder to each decoder
- Each encoder is composed of a self-attention component and a feed-forward NN
- Each decoder is composed of self-attention component, encoder-decoder attention component and a feed-forward NN





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

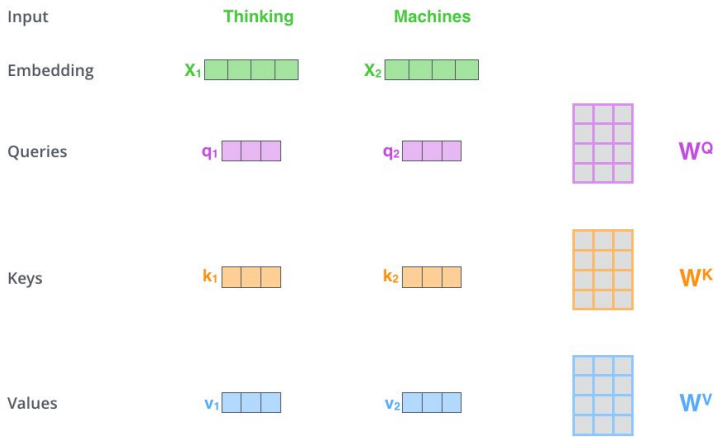
Self-attention in detail



https://arxiv.org/pdf/1706.03762.pdf

Query, key and value vectors

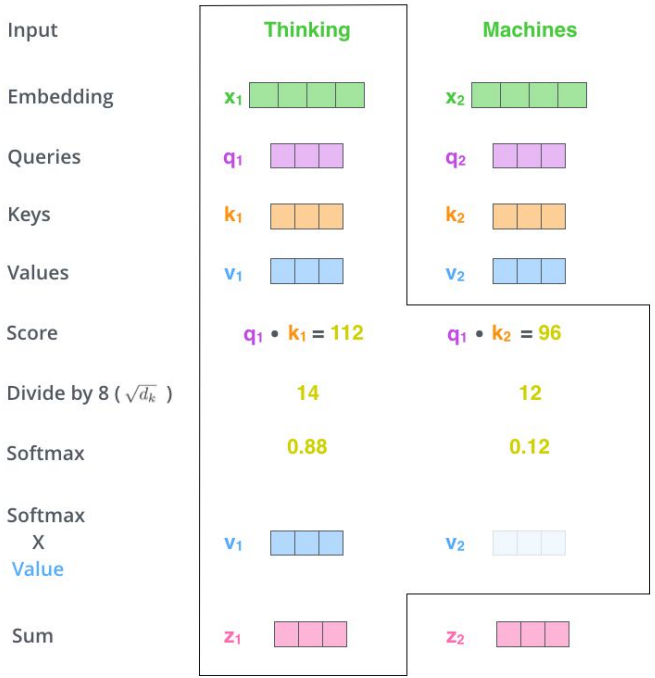
- Embedded input vectors are transformed into query, key and value vectors by multiplication with query, key and value weight matrices respectively



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

Scoring

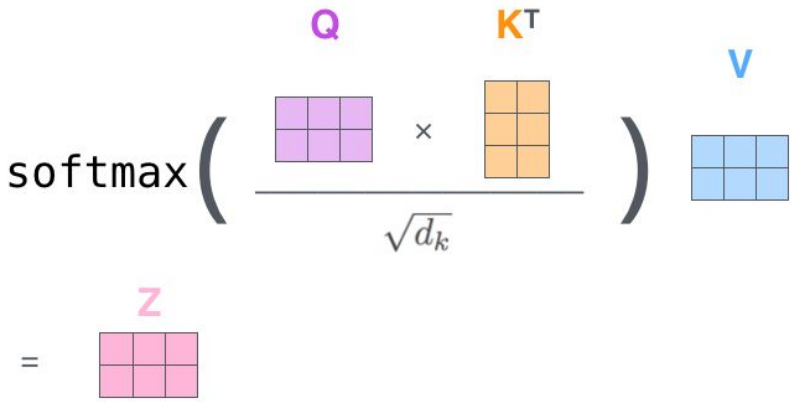
- Each word is then compared to every other word to obtain scores between all words by multiplying the query vectors with the key vectors and normalizing them
- Each score is then multiplied with the value vector (to diminish irrelevant words) and then all the value vectors are summed and sent to the next encoder



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

Attention as matrix multiplication

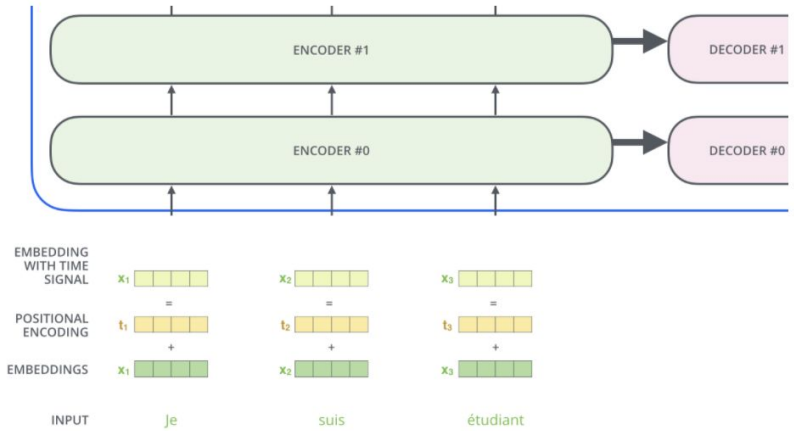
- Q^*K^T to produce scores
- Normalization of scores
- Softmax to get probabilities
- Multiply by V to get final score



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

Positional encoding

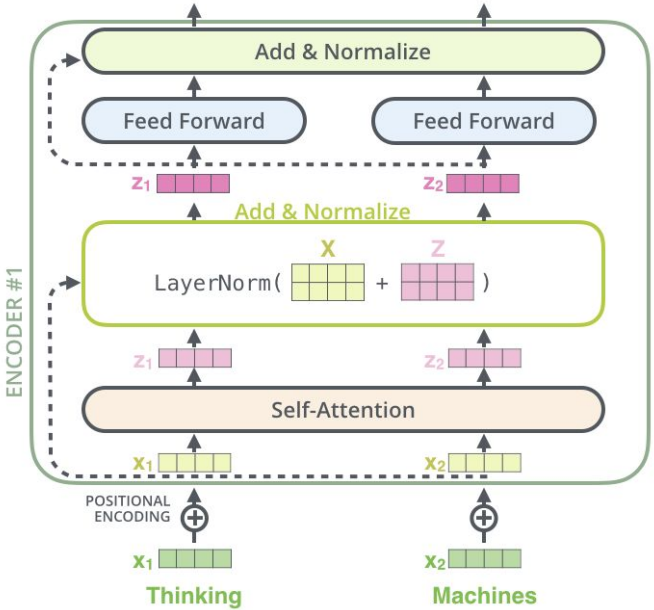
- We need to describe the position of each word in the sentence
- The Transformer adds a vector to each word's embedding
  - The vector follows a certain pattern about its position that the model learns



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

Residuals

- Each sub-layer in each encoder has a residual connection around it
- Therefore, sub-layers are followed by layer normalization layers



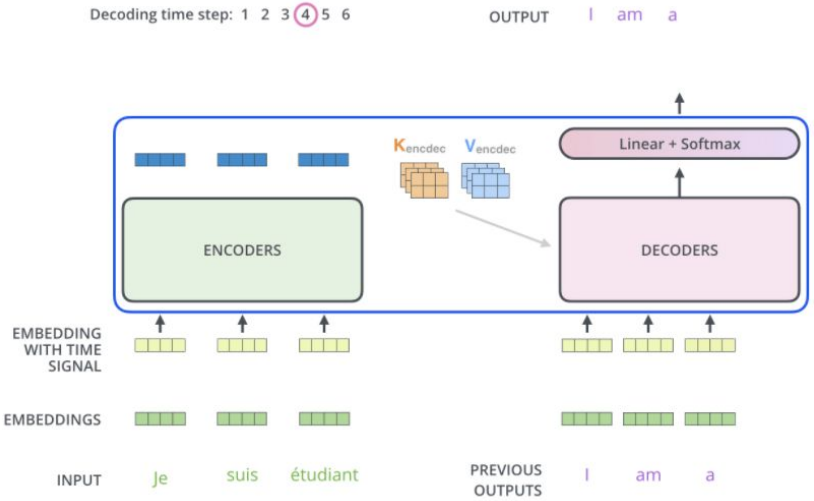
https://jalammar.github.io/illustrated-transformer/

Layer Normalization

- Extension of Batch Normalization
- Batch Normalization is theoretically done on the mean and standard deviation of the whole dataset, which is impractical
- Instead, in practice the current mini-batch is used as an estimator of the mean and standard deviation
- Which puts constraint on the size of the mini-batch
- Layer Normalization overcomes the drawbacks by computing the mean and standard deviation on the current layer

Decoding

- Decoding works in a very similar way to encoding
- The last encoder passes its output to each decoder
  - There, it serves as the key and value matrices
- Query matrix is the output of the decoder one layer below
- Self-attention in the decoders are only allowed to attend to earlier position in the output sequence (which is solved with masking out the illegal connections)



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

Converting outputs into a word

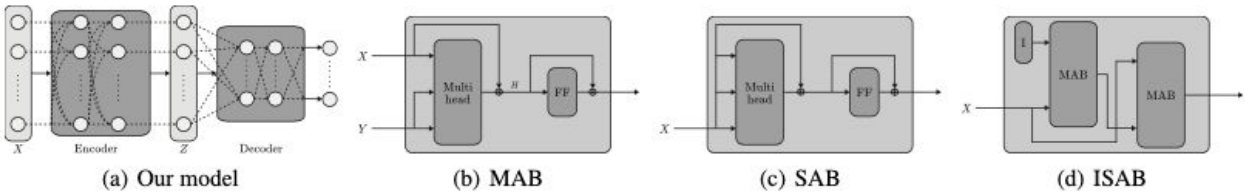
- Final layer after encoding and decoding is a feed-forward NN with a softmax layer
- The NN projects the output vectors into a logits vector that represents the whole english (for example) vocabulary
- The softmax layer produces a probability for each logit to be the final word
- The final word is simply the one with the highest probability

The loss function

- Input and output are probability distributions
- Kullback-Leibler divergence used to compare both distributions
- Each probability distribution is represented by a vector of the vocabularies
- Each word has a different probability distribution
    - “I” will have a distribution with “I” having the highest probability
    - “Am” will have a distribution with “am” having the highest probability
- This is plausible because the words are predicted one by one

Set Transformer

- Set Transformer[10] adjusts Transformer for sets that are required to be permutation invariant
- Presents a number of Set Attention Blocks
- Presents a parametrized pooling function that can adapt to the problem at hand
- Doesn't use positional encoding



https://arxiv.org/pdf/1810.00825.pdf

Multi-head Attention Block

- MAB(X, Y)
- Performs Multi-head attention between two input vectors and learnable parameters
- First vector is used as the query vector and second vector as both key and value vectors

$$\mathrm{MAB}(X, Y) = \mathrm{LayerNorm}(H + \mathrm{rFF}(H)), \tag{6}$$

$$\text{where } H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X, Y, Y; \omega)), \tag{7}$$

https://arxiv.org/pdf/1810.00825.pdf

Set Attention Block

- SAB(X) = MAB(X, X)
- Basically performs self-attention on given set X
- The output of SAB contains information about pairwise interactions among the elements of X
- Therefore, multiple SABs can be stacked to encode higher order interactions
- However, SAB has a quadratic complexity, which could be a problem for large sets

Induced Set Attention Block

- ISAB(X) = MAB(X, MAB(I, X))
- Inducing points I are trainable parameters of lower dimension
- MAB(I, X) attends the inducing points to the input set X
- Then again the first set is ran through MAB with X to produce the result
- The idea is to project X into a lower dimension, and then reconstruct it
- Unlike SAB's O(n^2) complexity, ISAB has O(nm) complexity, where m is typically a small number

Pooling by Multihead Attention

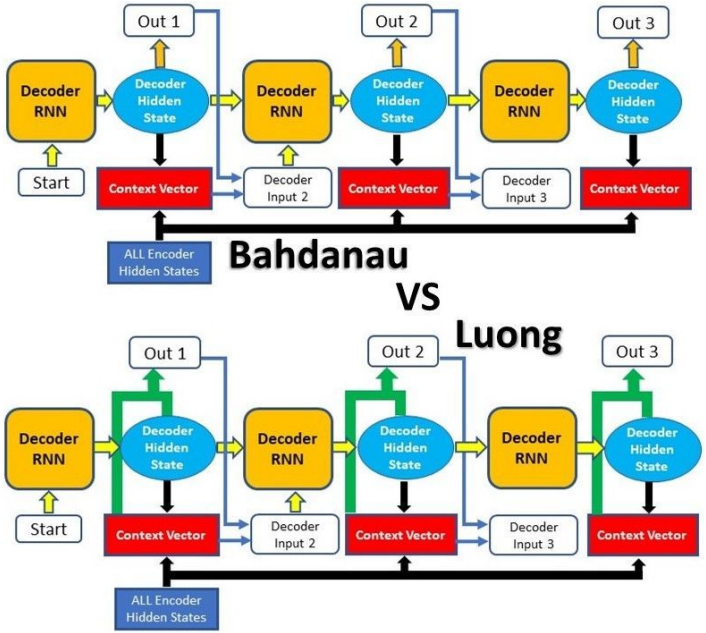
- $PMA_k(Z) = MAB(S, rFF(Z))$
- S is a set of k learnable seed vectors
- Output of PMA is a set of k items
- Those k seed vectors are then ran through SAB to further model the interactions
- $SAB(PMA_k(Z))$
- This has been shown only empirically that it increases the performance

Broad categories of attention mechanism

- Self-attention[7]
  - Relating different positions of the same input sequence
- Global/Soft[8]
  - Attending to the entire input state space
- Local/Hard[9]
  - Attending to a part of the input state space (e.g. a part of an image)
  - Predicts a single position and then computes a context vector in a window around that position

Popular attention mechanisms

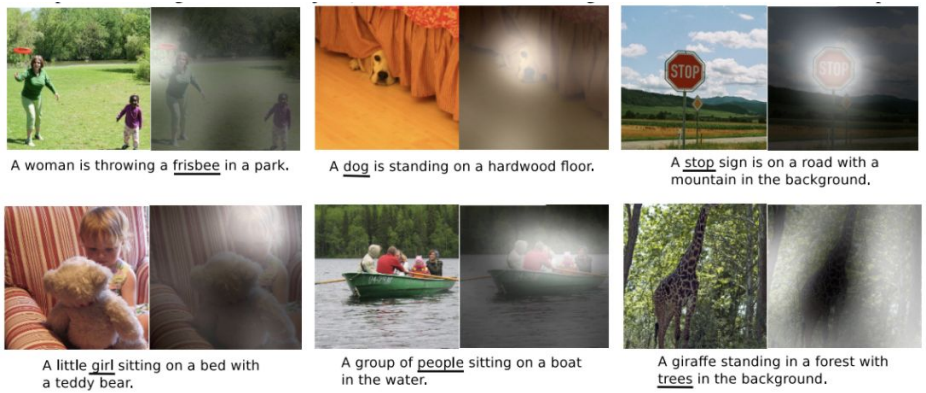
- General: score(s_t, h_i) = $s_t^T W_a h_i$ where W_a is a trainable weight matrix
- Dot-Product: score(s_t, h_i) = $s_t^T h_i$
- Scaled Dot-Product: score(s_t, h_i) = $s_t^T h_i$ / √n
- Content-based attention: score(s_t, h_i) = cos($s_t^T h_i$)



https://blog.floydhub.com/attention-mechanism/

Application: Image Captioning

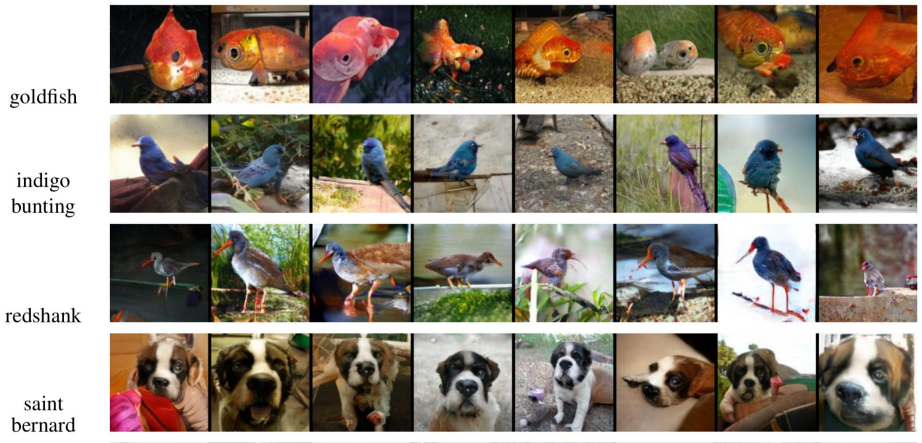
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
- Attempts to align the input image and output word
- Captions are aligned with specific parts of the image, highlighting the relevant objects



https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#born-for-translation

Application: Image Generation

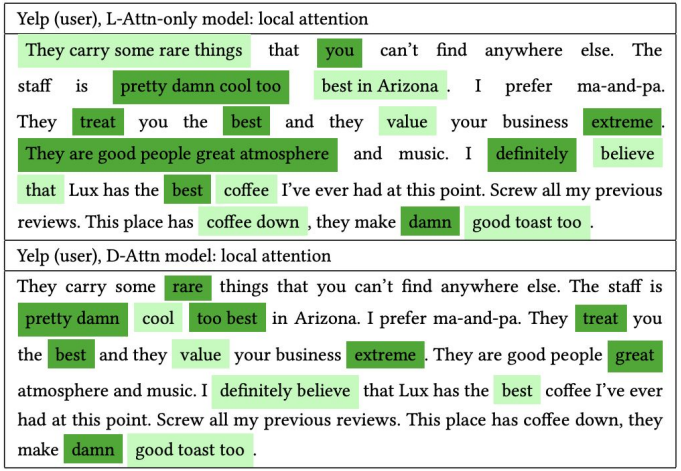
- Self-Attention Generative Adversarial Networks



https://arxiv.org/pdf/1805.08318.pdf

Application: Sentiment Analysis/Recommender Systems

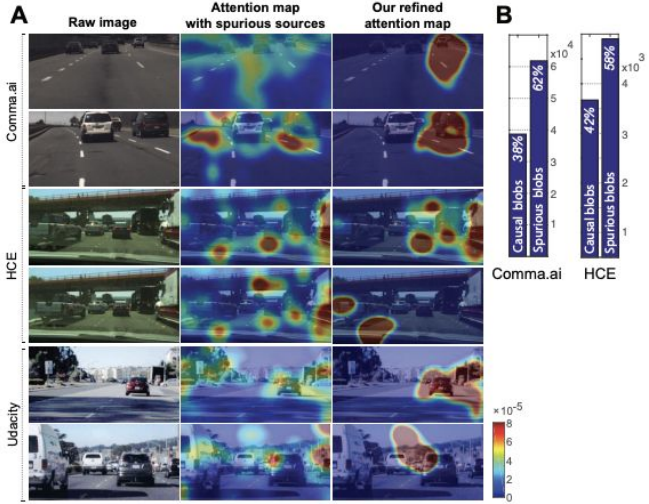
- Interpretable Convolutional Neural Networks with Dual Local and Global Aention for Review Rating Prediction
- Predicts importance of a ward in a sentence
- From that, builds up a recommender system



https://dl.acm.org/doi/10.1145/3109859.3109890

Application: Self-driving cars

- [Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention]
- Uses local attention to spot places of a self-driving car's interest



https://arxiv.org/pdf/1710.10903.pdf

Demo

https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC

Further reading

- https://jalammar.github.io/illustrated-transformer/
- https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
- https://buomsoo-kim.github.io/attention/2020/01/01/Attention-mechanism-1.md/
- https://blog.floydhub.com/attention-mechanism/
- https://icml.cc/media/Slides/icml/2019/halla(10-09-15)-10-15-45-4343-a_tutorial_on.pdf

References

[1] Sequence to Sequence Learning with Neural Networks

[2] Attention Is All You Need

[3] Attention-Based Models for Speech Recognition

[4] Graph Attention Networks

[5] Multi-Pointer Co-Attention Networks for Recommendation

[6] Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention

[7] Long Short-Term Memory-Networks for Machine Reading

[8] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

[9] Effective Approaches to Attention-based Neural Machine Translation

[10] Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks

Thanks for your attention!