Attention and Transformers

Keith L. Downing

The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
keithd@idi.ntnu.no

March 1, 2023
University of Montreal


Google Brain (Commonly cited Transformer article)


Web Pages

- https://www.tensorflow.org/text/tutorials/transformer
- machinelearningmastery.com (Search for many keywords: attention, transformer, positional encoding, etc.)
While playing poker with my friends, I noticed a small mirror behind …

Effective translation cannot be done in a word-for-word manner

While playing poker with my friends, I noticed a small mirror behind …

Mens jeg spilt poker med mine venner, la jeg merke til et lite speil bak …

Mens jeg spilt poker med mine venner, la jeg merke til et lite speil bak …

Probability Distribution over all words in the lexicon

Compute gradients + Modify weights / biases in Encoder and Decoder

Teacher-Forced Training

RNNs and LSTMs process sequences sequentially and train via BPTT.

Transformers process text chunks in parallel. This requires more memory but generates shorter derivative chains ⇒ fewer vanishing gradients.

Keith L. Downing

Attention and Transformers
While playing poker with my friends, I noticed a small mirror behind …

The vector representation for each word = a contextualization of that word within some large corpus

Downstream processing

Attention

Embedding

While playing poker with my friends, I noticed a small mirror behind …

The vector representation for each word = a contextualization of that word within both the large corpus and the current text chunk.

Downstream processing

Attention

Local Interactions

Attention ↔ Embedding vectors interact and change, thus adding local context to the representations sent downstream.

Left: Standard approach to embedding.
Left + Right: Standard embeddings enhanced with attention.
Attention ↔ Embedding vectors interact and change, thus adding local context to the representations sent downstream.
Attention \approx \text{Fuzzy Database Lookup}

### Scaled Dot-Product Attention

\[
\text{Attention}(Q, K, V) = \sum_i [\text{similarity}(Q, K_i) \cdot V_i] = \text{Softmax}\left( \frac{Q \cdot K^T}{\sqrt{d}} \right) \cdot V
\]

- **Q** = Query, **K** = Keys, **V** = Values, **d** = dimensions of **Q** and **K_i**

Keith L. Downing

Attention and Transformers
Attention as Context

Attention-driven behavior, e.g., in a transformer

- Input a chunk of text, with each of k words (W) represented by its embedding (a vector of length m), bundled into a k x m matrix, V.
- Each row, $V_i$ represents the original semantics of word $W_i$, found using any of the standard methods for computing word embeddings.
- The goal is to re-represent each $W_i$ with respect to its context in the current text chunk.
- In this case, $V_i$ = ith query, the DB keys = V, and the DB values = V.
- Then, contextualized $W_i$ is simply $\text{attention}(V_i, V, V)$
- The fully contextualized text is a matrix C, with row $C_i = \text{attention}(V_i, V, V)$
- C is then sent downstream in the neural network.
- Visualizations of attention often involve a heatmap matrix, H. In this, $H_{i,j} = V_i \cdot V_j$ = the similarity between embedding vectors $V_i$ and $V_j$, where $V_i$ embodies the query, and $V_j$ the key.
- The $V_i$ are often pre-processed with positional encodings (see later).
- This is called self-attention, since V = the query and key matrix.
## Multiple Queries at Once

### Scaled Dot-Product Attention

Attention \((Q, K, V) = \text{Softmax}(\frac{Q \cdot K^T}{\sqrt{d}}) \cdot V\)

- **Q** = Queries, **K** = Keys, **V** = Values, \(d = 2nd\) dimension of **Q** and **K**

---

Keith L. Downing

Attention and Transformers
def similarities(queries, keys, scale=True):
    m = numpy.dot(queries, keys.transpose())
    d = queries.shape[1]
    if scale: m = m / math.sqrt(d)
    return numpy.array([P1.softmax(row) for row in m]).reshape(queries.shape[0], d)

def attention(queries, keys, values, scale=True):
    sims = similarities(queries, keys, scale=scale)
    return numpy.dot(sims, values)

def self_attention(embeddings, scale=True):
    return attention(embeddings, embeddings, embeddings, scale=scale)
Graph Models of Attention

Each of these steps is easily differentiable.

Use mask when some of the values should not be considered, e.g., in parts of a transformer's decoder module.

Based on Attention Is All You Need, 2017.
Multi-Head Attention

- Each triple of the Q,K,V weights is an alternate transformation of Q,K and V.
- Each transformation is often of lower dimension than Q,K, and V embeddings.
- Similar to use of multiple channels in convolution nets.

Based on *Attention Is All You Need*, 2017.
**Self-Attention**: Queries and Key-Value pairs have same source.

**Attention**: Queries and Key-Value pairs may have different sources.

The above English and Norwegian texts have unusual composition in order to make the sentences interesting but short.
Attention Modules and Masks in a Transformer

**Training Phase**

**Target = “Henry”**

**Translator**

Although rotten Henry ate his apple

This text segment is entered all at once

When contextualizing these three words, we cannot include keys/values for Norwegian text that comes later in the sequence, since that is what we are trying to predict.

=> The similarity matrix must be MASKED when computing self-attention.

These are actually entered all at once also, but it helps to view them sequentially.

**Current Input**

**Selv råtne spiste**

**Next Data Case**

**Selv råtne spiste**

**Henry**

**Target = “eple”**

**When contextualizing these three words, we cannot include keys/values for Norwegian text that comes later in the sequence, since that is what we are trying to predict.**

=> The similarity matrix must be MASKED when computing self-attention.

**When contextualizing these three words, we cannot include keys/values for Norwegian text that comes later in the sequence, since that is what we are trying to predict.**

=> The similarity matrix must be MASKED when computing self-attention.

Language Translation Transformer

Masked Self-Attention HeatMap

Note: Self-attention for the English text does not need masking, since context for the source language is **bidirectional**, but only **uni-directional** for the target language.
Transformer Training

Decoder's Input Matrix

- < Start Text Marker >
- Selv
- råtne
- spist
- Henry
- eple
- sitt

“Shifted Right”

Embedding Vectors

Complete Source Text

Self Attention

Encoder

Decoder's Target Outputs

- Selv
- råtne
- spist
- Henry
- eple
- sitt

< End Text Marker >

One-hot Vectors

“Shifted Right”

Predicted Word #1

Predicted Word #2

…

…

…

…

Predicted Word #7

Error

Q

K

V

0's Mask

Masked Self-Attention

Decoder

Contextualized Source Text

Self Attention

Complete Source Text

Gradients

Self Attention

Complete Source Text

Encoder

Decoder

Keith L. Downing

Attention and Transformers
Standard Transformers have 2 copies of the weight matrix and RELU, in series, in both encoder and decoder modules.
Since attention modules handle chunks of input in parallel, there is no positional information (neither explicit nor implicit) in the data.

Solution: supplement each element of each embedding vector with a specially-designed code for position: $P(\text{sequence \_index}, \text{embedding \_index})$.

The scheme from "Attention Is All You Need" is trigonometric:

$$P(k, i) = \sin\left(\frac{k}{n^d}\right) \quad \text{for even } i$$

$$P(k, i) = \cos\left(\frac{k}{n^d}\right) \quad \text{for odd } i$$

where $d =$ dimension (length) of embedding vectors, $i =$ index within an embedding, $k =$ index of embedding in sequence, and $n = 10000$.

$\forall k, i : \text{Embedding}(k, i) \leftarrow \text{Embedding}(k, i) + P(k, i)$
Sinusoidals for Positional Encoding

\[ P(k, i) = \sin\left(\frac{k}{n^d}\right) \]

- \( n = 10,000; \ d = 512 \)
- frequency increases as \( k \uparrow \) but decreases as \( i \uparrow \).
Positional Encoding: Why so Complex?

- The goal is for each word to have a different signature, but there is no need for every embedding unit to have one.
- However, if some unique code were only assigned to each word, and then given to every embedding unit in the word, then:
- Due to the frequency of normalizations (e.g. softmax) in NN's, the shifting effect can disappear (see below)
- Hence, each word needs a unique pattern of position assignments to its embedding units.
- Since each value of k produces a different sinusoid, each word embedding has different relationships between the units of its embedding.
- Words with nearby locations have similar such relationships ⇒ more likely to produce high similarity weights.

Why shared increments don’t work well

- Let $W_i$ and $W_j$ be the same word, with embedding = $(0.1, 0.2, 0.3)$, that appears twice in the same text chunk.
- Give positional increments of 0.1 and 0.7 to $W_i$ and $W_j$, respectively.
- $W_i^* = (0.2, 0.3, 0.4)$ and $W_j^* = (0.8, 0.9, 1.0)$
- $\text{Softmax}(W_i^*) = (0.3, 0.33, 0.37) = \text{Softmax}(W_j^*)$ ⇒ All positional info is lost!
OpenAI’s Systems

- Parameters (weights + biases): GPT-2: 1.5 billion; GPT-3: 175 billion
- Training set size: GPT-2: 40 GB; GPT-3: 570 GB
- Estimated cost to train GPT-3: $4.6 million!!
- ChatGPT
  - Uses Supervised Fine-Tuning (SFT) model:
  - SFT = GPT-3 + Human-generated cases (for tasks such as question answering). This supervision is the fine tuning.
  - 40 people produced 13,000 cases.
  - Also uses Proximal Policy Optimization (PPO), a form of RL, with rewards based on user preferences when presented with several options by the network.