UNIVERSITY of LOUISIANA LAFAYETTE

CSCE566-DATA MINING
WEEK 6

Time-Series Data Mining: Transformer

Min Shi
min.shi@louisiana.edu

Oct 2, 2024

Recap

Altaba Inc. Stock Price Prediction
training
Actual Stock Price
Predicted Stock Price
Altaba Inc. Stock Price
Time
1996
2000
2004
2008
2012
2016

My experience so far has been fantastic!
POSITIVE
The product is okay I guess.
NEUTRAL
Your support team is useless.
NEGATIVE

Actual_4h
Actual_8h
Actual_12h
Actual_16h
Actual_20h

Recap

Goal: Model Sequential Data (Recall RNN)

Each hidden state is a function of the previous hidden state

Recap

Encoding of the source sentence.
Provides initial hidden state
for Decoder RNN.
Target sentence (output)
Encoder RNN
Decoder RNN
argmax
the poor don't have any money <END>
les pauvres sont démunis
<START> the poor don't have any money
Source sentence (input)

Recap

Problem: RNNs Use Sequential Computation

Seemingly hard for RNNs to carry information through hidden states across many time steps and train/testing is slow

Recap

Idea: Model Sequential Data Without Recurrence

Replace sequential hidden states for capturing knowledge of other inputs with a new representation of each input that shows its relationship to all other inputs (i.e., self-attention)

Models for time-series data mining

RNN

LSTM

GRU

Transformers

Transformer

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.

Attention

- This can also help us understand the network's mistakes.

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Arrow thickness is indicative of attention weight

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

A large attention score means the other word will strongly inform the new representation of the word

Transformer Intuition

What does bank mean in this sentence?

I arrived at the bank after crossing the …

Transformer Intuition

What does bank mean in this sentence?
- the new representation of the word disambiguates the meaning by identifying other relevant words (e.g., high attention score with “river”)

I arrived at the bank after crossing the river

vs

I arrived at the bank after crossing the street

Transformer vs RNN (Intuition)

I arrived at the bank after crossing the …

…street? …river?

What does **bank** mean in this sentence? Meaning depends on other input words

Transformer vs RNN (Intuition)

I arrived at the bank after crossing the …

…street? …river?

What does **bank** mean in this sentence? Meaning depends on other input words

I've no idea: let's wait until I read the end

RNNs

O(N) steps to process a sentence with length N

I don't need to wait – I see all words at once!

Transformer

Constant number of steps to process any sentence

Transformer: A Suggested Definition

"Any architecture designed to process a connected set of units—such as the tokens in a sequence or the pixels in an image—where the only interaction between units is through self-attention."

Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens

https://towardsdatascience.com/self-attention-5b95ea164f61

Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Rashonda accepted a job in deep learning because she loves the topic

And so on for remaining words…

Self-Attention: Disambiguates Word Meanings

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Rashonda accepted a job in deep learning because she loves the topic

A better representation of “she” would encode information about “Rashonda”

Self-Attention: Disambiguates Word Meanings

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

I arrived at the bank across the river

A better representation of “bank” would encode information about “river”

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Self-Attention vs General Attention

Self-attention

Relates tokens from the same source

General attention

Relates tokens from different sources

Computing Self-Attention

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\text{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$$

"How relevant is source token k for target step t?"

Attention input

$s_1, s_2, \ldots, s_m$ — all encoder states

h_t — one decoder state

Key difference 2: attention score multiplied with a value derived from the input

Key difference 1: input for self-attention

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Computing Self-Attention: Example

New representation of each input token to reflect each one's relationship to all tokens

Input tokens

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

- How many inputs are in this example?
- What is each one's dimensionality?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Three vectors are derived for each input by multiplying with three weight matrices (learned during training): query, key, and value

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., key weights

```
[0, 0, 1]
[1, 1, 0]
[0, 1, 0]
[1, 1, 0]
```

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., value weights

```
[0, 2, 0]
[0, 3, 0]
[1, 0, 3]
[1, 1, 0]
```

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., query weights

```
[1, 0, 1]
[1, 0, 0]
[0, 0, 1]
[0, 1, 1]
```

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

How many weight matrices are learned in this example?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Why do we learn the three weight matrices?

For each input, 2 of the derived vectors are used to compute **attention weights** (query and key) and the 3rd is **information** passed on for the new representation (value)

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

We now will examine how to find the new representation for the first input.

Computing Self-Attention: Example

Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

1	0	2

x

0
1
1

= ?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

1	0	2

x

4
4
0

= ?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

1	0	2

x

2
3
1

= ?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Why dot product? Indicates similarity of two vectors

- Match = 1 (i.e., cos(0))
- Opposites = -1 (i.e., cos(180))

https://towardsdatascience.com/self-attention-5b95ea164f61

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Can also use similarity measures other than the dot product

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention weights: softmax scores for all inputs to quantify each token's relevance; e.g.,

= softmax([2, 4, 4])

= [0.0, 0.5, 0.5])

To which input(s) is input 1 most related?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values

2. Sum all weighted vectors

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention weights amplify input representations (values) that we want to pay attention to and repress the rest

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 2 most related?`

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
   - Softmax resulting 3 scores from query x keys

2. Compute weighted sum of values using attention scores

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 3 most related?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

2. Compute weighted sum of values using attention scores

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Efficient Computation for Self-Attention

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

Efficient Computation for Self-Attention

Implementation detail: scaling down the size helps preserve gradients needed for training; *k* is dimensionality of the key vector

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

Self-Attention vs RNN

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

https://towardsdatascience.com/self-attention-5b95ea164f61

Multi-head Attention

- **Goal**: enable each token to relate to other tokens in multiple ways
- **Key idea**: multiple self-attention mechanisms, each with their own key, value and query matrices

https://sebastianraschka.com/pdf/lecture-notes/stat453ss21/L19_seq2seq_rnn-transformers____slides.pdf

Multi-head Attention

1) Create query, key, and value vectors for all attentions heads

2) Compute new input representations

3) Condense all representations into a single representation by concatenating **z**-s and multiplying by a weight matrix

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

Trained multi-head attention examples

Figure shows two columns of attention weights for the first two attention heads

- Darker values signify larger attention scores

What does “it” focus on most in the first attention head?

- The animal (e.g., represents what is “it”)

What does “it” focus on most in the second attention head?

- tired (e.g., represents how “it” feels)

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

Trained multi-head attention examples

Figure shows five columns of attention weights for five attention heads

- Darker values signify larger attention scores

Attention weights may be hard to interpret

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

Transformer: Architecture

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

[PDF] **Attention** is **all you need**

A Vaswani - Advances in Neural Information Processing Systems, 2017 - user.phil.hhu.de

Attention is **all you need Attention** is **all you need** …

☆ Save 99 Cite Cited by 135017 Related articles »

Transformer: Architecture

Transformer: Architecture

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Self-attention (in encoder)

Transformer: Architecture

Transformer: Architecture

Transformer: Architecture

Transformer: Architecture

Multi-head self-attention

Transformer: Architecture

Position embeddings are *added* to each word embedding. Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!

Transformer: Architecture

Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow

Transformer: Architecture

A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity

Transformer: Architecture

We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!)

Transformer: Architecture

Moving onto the decoder, which takes in English sequences that have been shifted to the right (e.g., *<START> schools opened their*)

Transformer: Architecture

We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

Transformer: Architecture

We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

Why don't we do masked self-attention in the encoder?

Transformer: Architecture

Now, we have *cross attention*, which connects the decoder to the encoder by enabling it to attend over the encoder's final hidden states.

Transformer: Architecture

After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word

Positional encoding

Positional encoding

Intuitive example

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

Transformer positional encoding

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

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}})$$

Positional encoding is a 512d vector
i = a particular dimension of this vector
pos = dimension of the word
d_model = 512

Transformer positional encoding

What does this look like?

(each row is the pos. emb. of a 50-word sentence)

Email: min.shi@louisiana.edu