Stony Brook University

Audio Analysis

Yongjun Zhang, Ph.D.
Dept of Sociology and IACS
https://yongjunzhang.com

Today's Agenda

1. Mini Lecture (6:00-6:50PM)
2. Lab Tutorial (7:20-8:20 PM)

Recurrent Neural Network

Disclosure: this tutorial was inspired by
Colah's blog
https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Raghav's blog
https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0

Speech Recognition

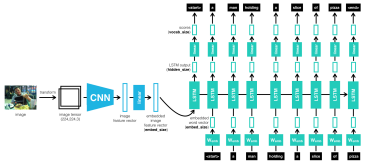
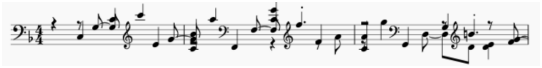


Image Caption

‹Google›1 , headquartered in ‹Mountain View›2 (‹1600 Amphitheatre Pkwy, Mountain View, CA›12 ‹1600›15 ‹Amphitheatre Pkwy›7 , ‹Mountain View›2 , ‹CA 940430›8 ‹940430›14), unveiled the new ‹Android›3 ‹phone›5 for ‹$799›13 ‹799›16 at the ‹Consumer Electronic Show›11 . ‹Sundar Pichai›4 said in his ‹keynote›9 that ‹users›6 love their new ‹Android›3 ‹phones›10 .

Name Entity Recognition



Music Generation

Amazon Customer

★★★★★ **Good Conceptual & Technical Guide**

Reviewed in the United States on February 18, 2016

Ansel Adams, as with any artist, cannot be successfully imitated, but if one can truly grasp and employ the concepts that drove Adam's genius, then, if interested in being a better photographer, they will succeed in excelling in photography.

This book takes one through forty of his photographs documenting in his words his thoughts, his actions and the equipment he used. in my opinion this is a helpful technical book regarding the circumstances surrounding the production of a great artist's works. If you are looking for wonderful

Sentiment Analysis

The Basic RNN Architecture

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

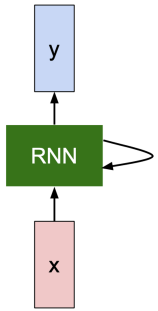
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

some function with parameters W



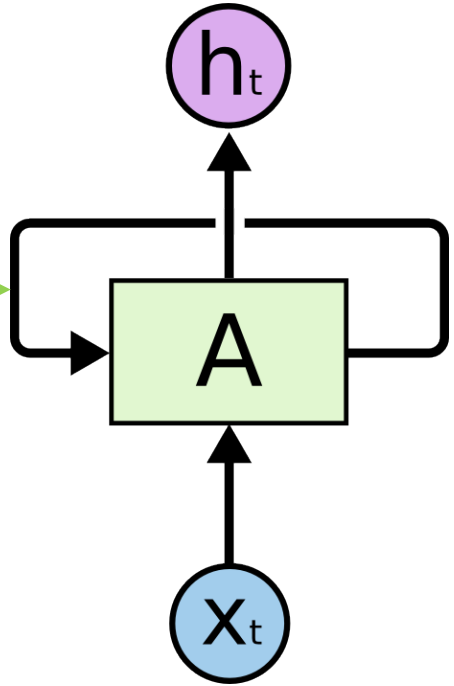
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

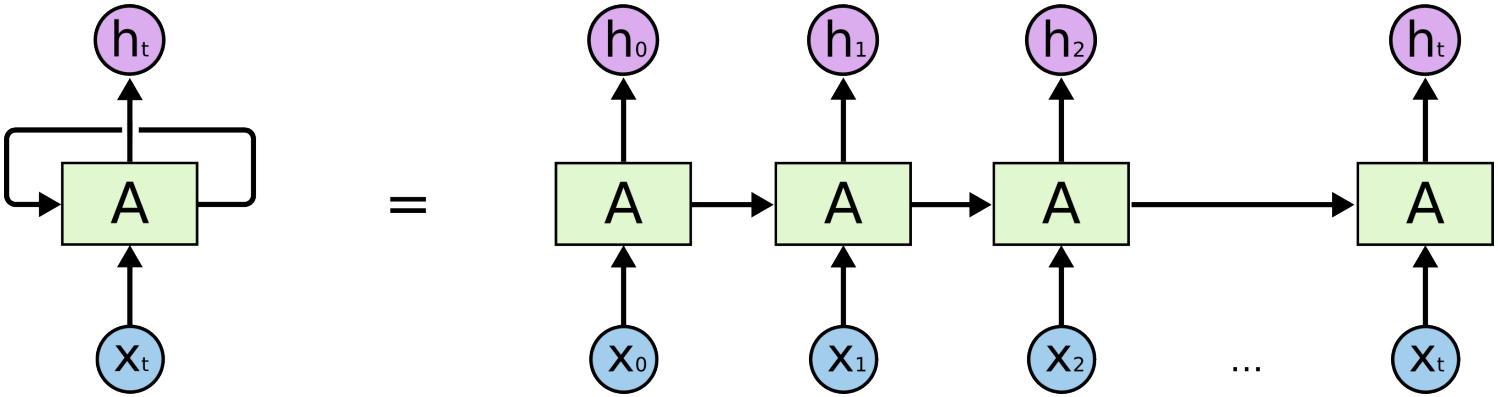
https://cs231n.github.io/

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$



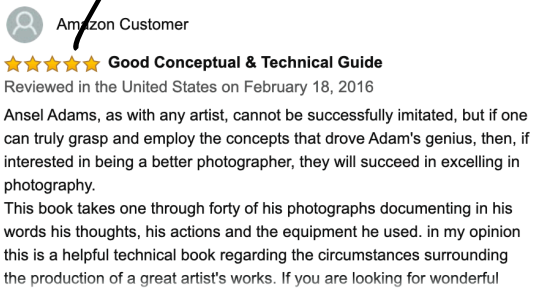
https://colah.github.io/

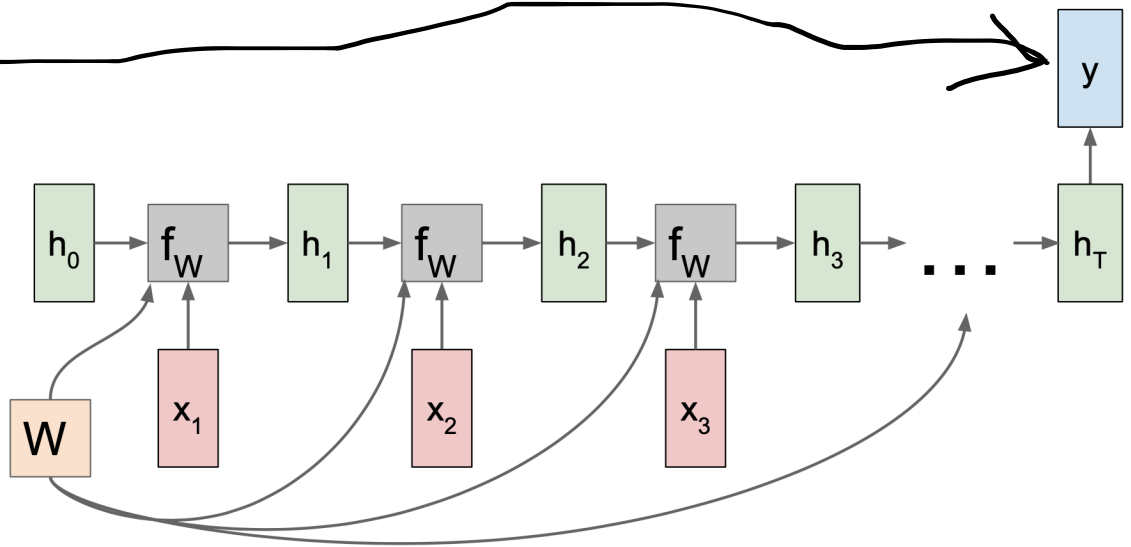


https://colah.github.io/

Let us take sentiment analysis as an example

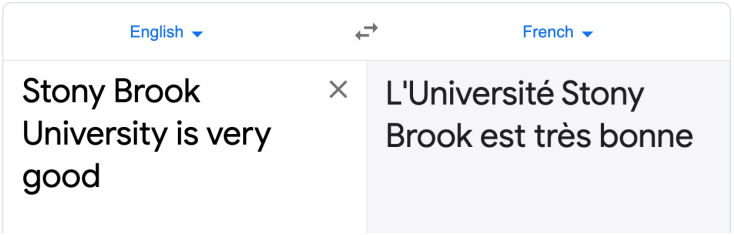
5 star?

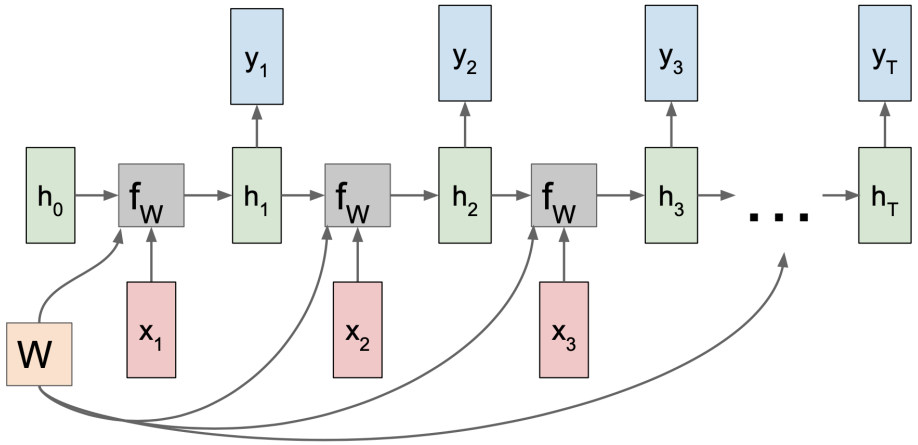




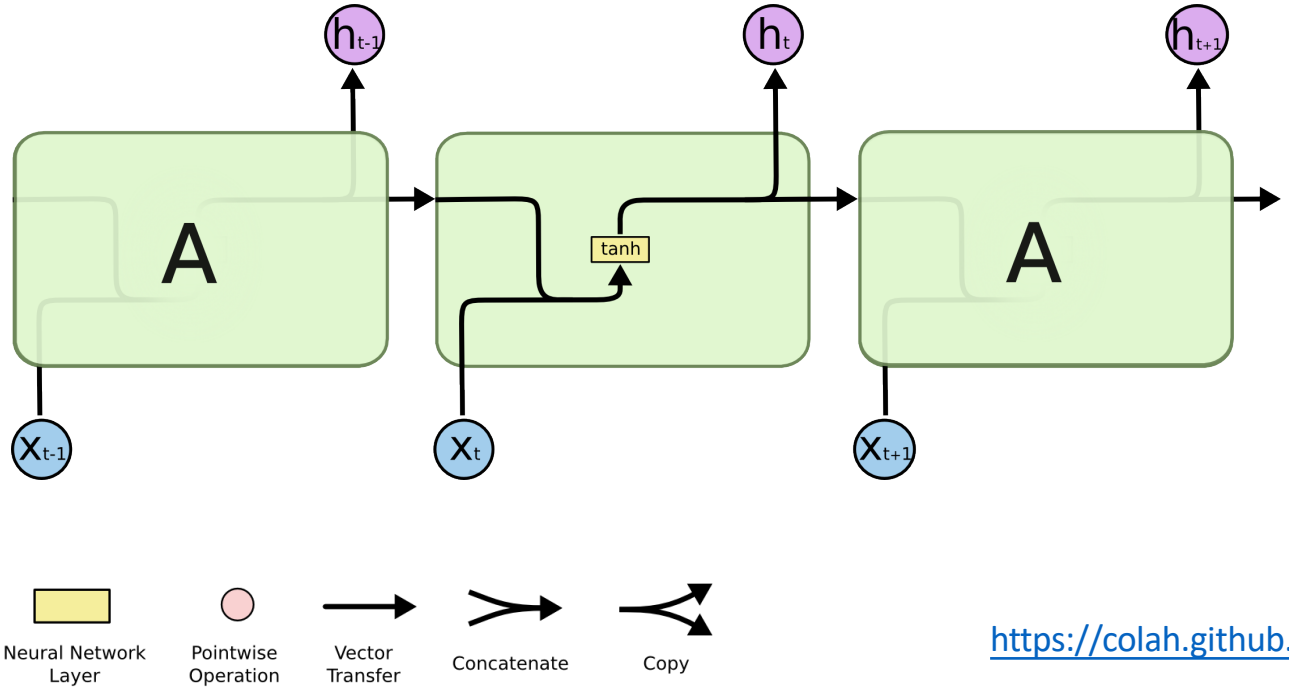
https://cs231n.github.io/

Let us take machine translation as an example



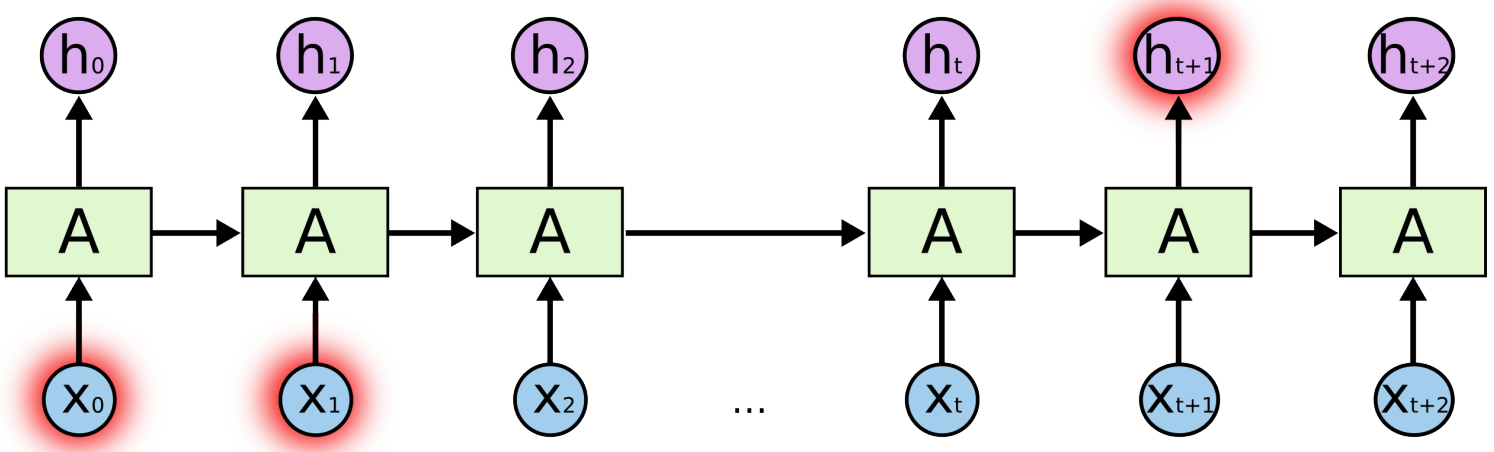


https://cs231n.github.io/

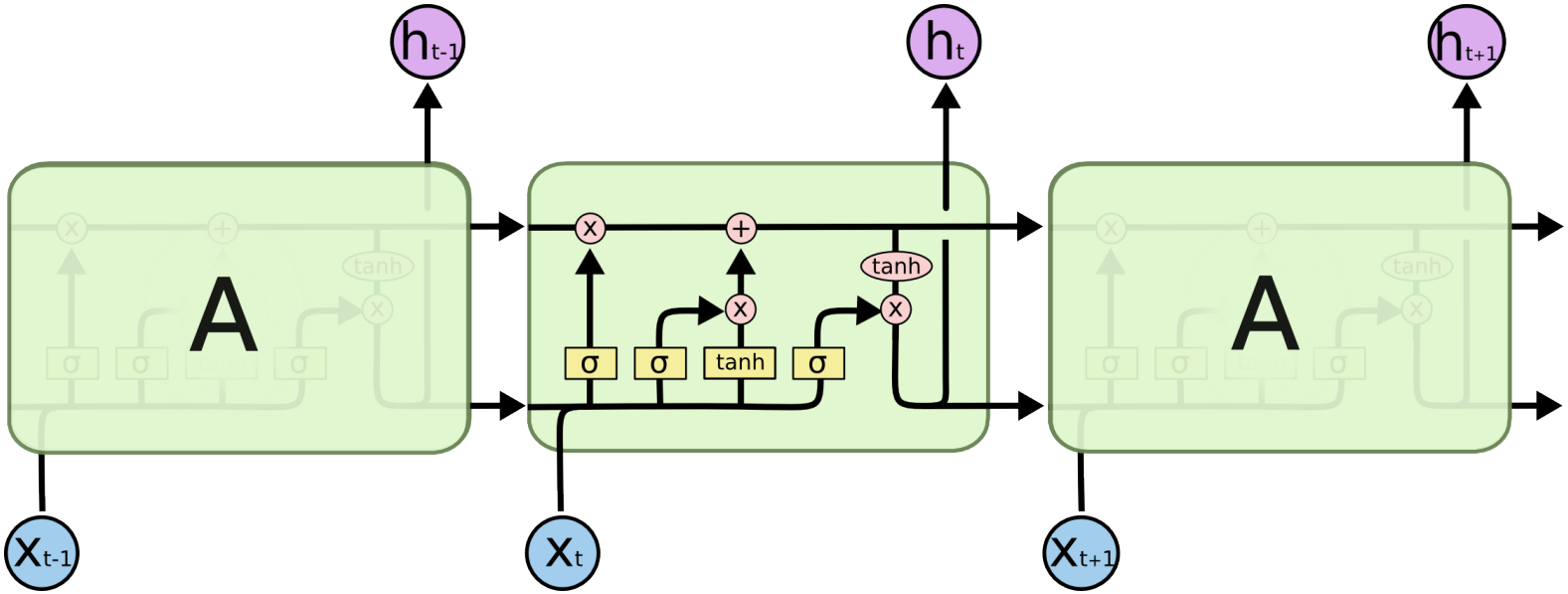


https://colah.github.io/

Memory loss and Gradient Vanishing

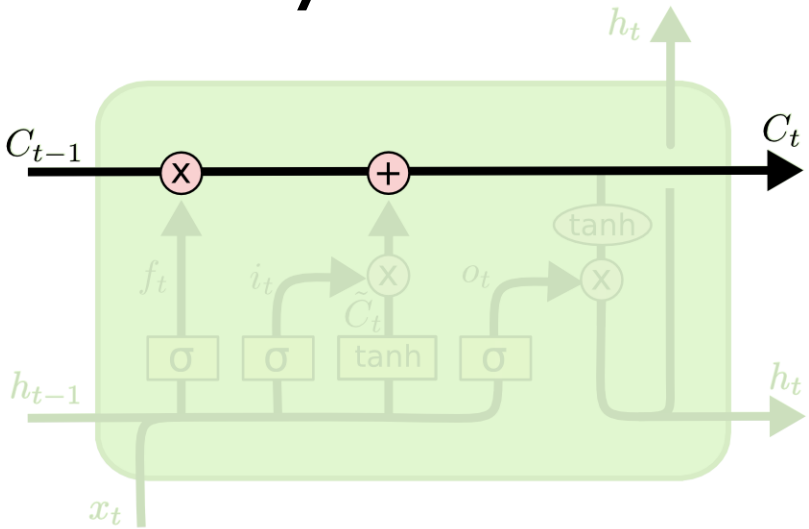


https://colah.github.io/



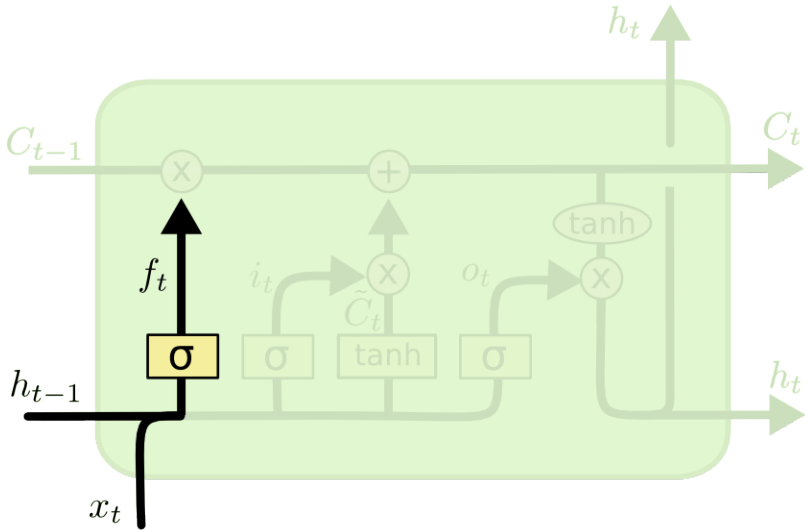
https://colah.github.io/

The key to LSTMs is the cell state, which allows information to flow easily.



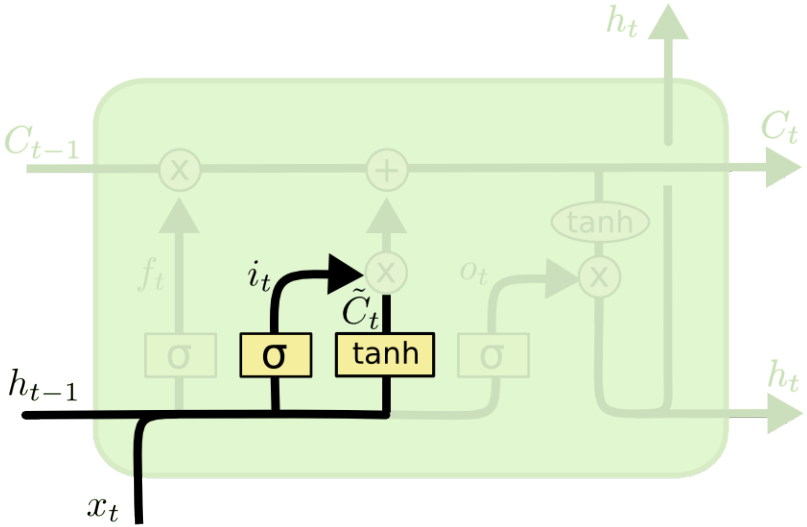
The LSTM can remove or add information to the cell state, regulated by gates.

https://colah.github.io/



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

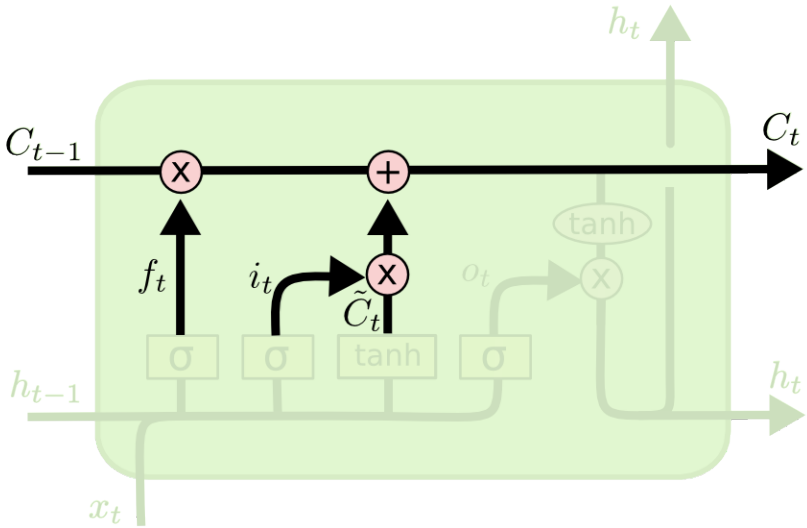
https://colah.github.io/



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

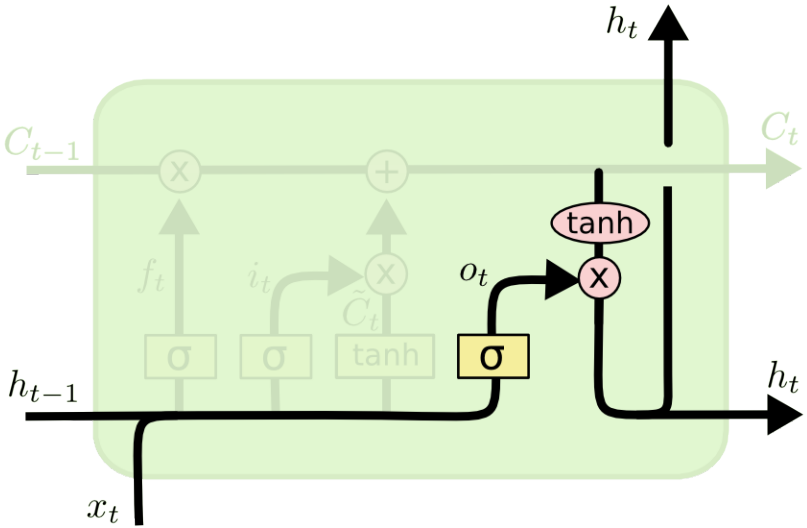
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

https://colah.github.io/



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

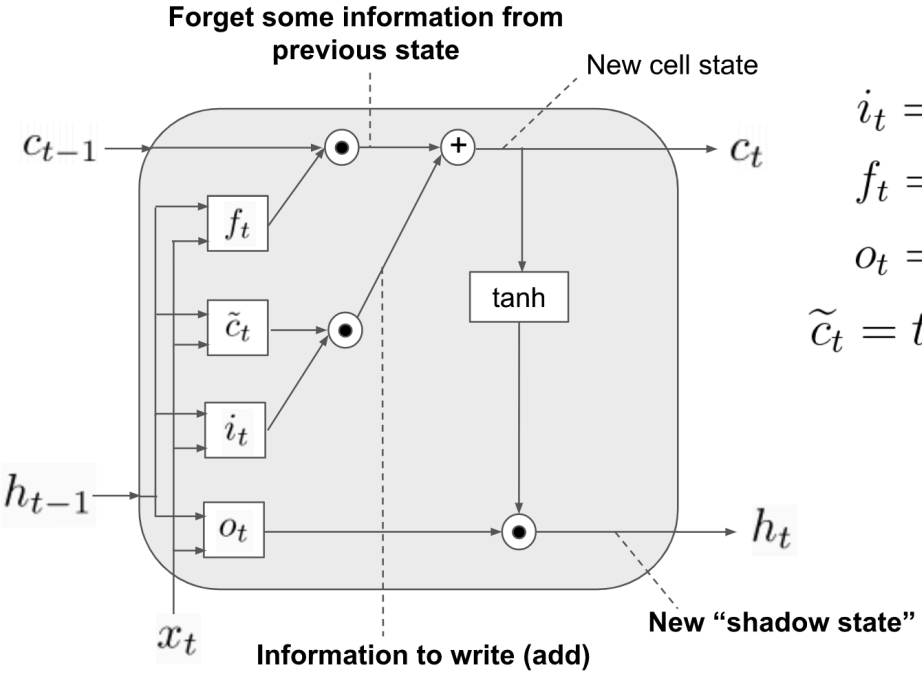
https://colah.github.io/



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

https://colah.github.io/



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

Deepsystems.ai

LSTM layer

`LSTM` class

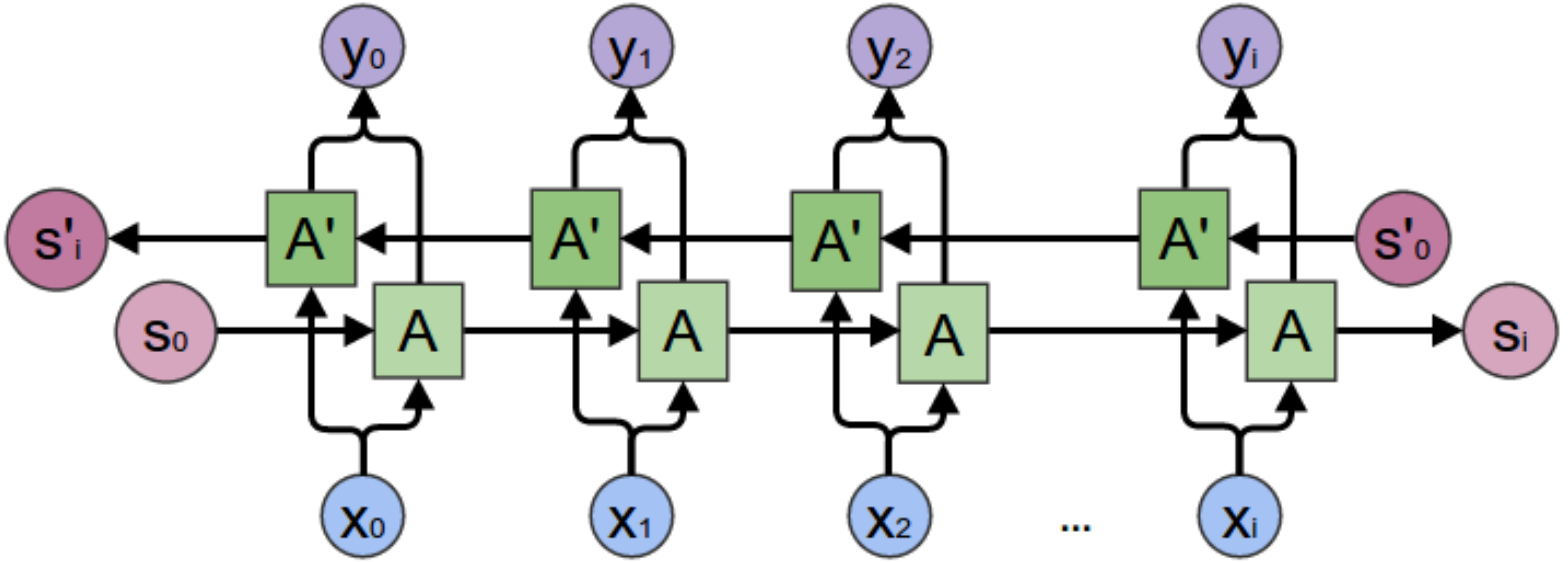
```
tf.keras.layers.LSTM(
    units,
    activation="tanh",
    recurrent_activation="sigmoid",
    use_bias=True,
    kernel_initializer="glorot_uniform",
    recurrent_initializer="orthogonal",
    bias_initializer="zeros",
    unit_forget_bias=True,
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    time_major=False,
    unroll=False,
    **kwargs
)
```

Long Short-Term Memory layer - Hochreiter 1997.

Arguments

- **units**: Positive integer, dimensionality of the output space.
- **activation**: Activation function to use. Default: hyperbolic tangent (`tanh`). If you pass `None`, no activation is applied (ie. "linear" activation: `a(x) = x`).
- **recurrent_activation**: Activation function to use for the recurrent step. Default: sigmoid (`sigmoid`). If you pass `None`, no activation is applied (ie. "linear" activation: `a(x) = x`).
- **use_bias**: Boolean (default `True`), whether the layer uses a bias vector.
- **kernel_initializer**: Initializer for the `kernel` weights matrix, used for the linear transformation of the inputs. Default: `glorot_uniform`.
- **recurrent_initializer**: Initializer for the `recurrent_kernel` weights matrix, used for the linear transformation of the recurrent state. Default: `orthogonal`.
- **bias_initializer**: Initializer for the bias vector. Default: `zeros`.
- **unit_forget_bias**: Boolean (default `True`). If True, add 1 to the bias of the forget gate at initialization. Setting it to true will also force `bias_initializer="zeros"`. This is recommended in Jozefowicz et al..
- **kernel_regularizer**: Regularizer function applied to the `kernel` weights matrix. Default: `None`.
- **recurrent_regularizer**: Regularizer function applied to the `recurrent_kernel` weights matrix. Default: `None`.
- **bias_regularizer**: Regularizer function applied to the bias vector. Default: `None`.
- **activity_regularizer**: Regularizer function applied to the output of the layer (its "activation"). Default: `None`.
- **kernel_constraint**: Constraint function applied to the `kernel` weights matrix. Default: `None`.
- **recurrent_constraint**: Constraint function applied to the `recurrent_kernel` weights matrix. Default: `None`.
- **bias_constraint**: Constraint function applied to the bias vector. Default: `None`.
- **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs. Default: 0.
- **recurrent_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state. Default: 0.
- **return_sequences**: Boolean. Whether to return the last output. in the output sequence, or the full sequence. Default: `False`.
- **return_state**: Boolean. Whether to return the last state in addition to the output. Default: `False`.
- **go_backwards**: Boolean (default `False`). If True, process the input sequence backwards and return the reversed sequence.
- **stateful**: Boolean (default `False`). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
- **time_major**: The shape format of the `inputs` and `outputs` tensors. If True, the inputs and outputs will be in shape `[timesteps, batch, feature]`, whereas in the False case, it will be `[batch, timesteps, feature]`. Using `time_major = True` is a bit more efficient because it avoids transposes at the beginning and end of the RNN calculation. However, most TensorFlow data is batch-major, so by default this function accepts input and emits output in batch-major form.
- **unroll**: Boolean (default `False`). If True, the network will be unrolled, else a symbolic loop will be used. Unrolling can speed-up a RNN, although it tends to be more memory-intensive. Unrolling is only suitable for short sequences.

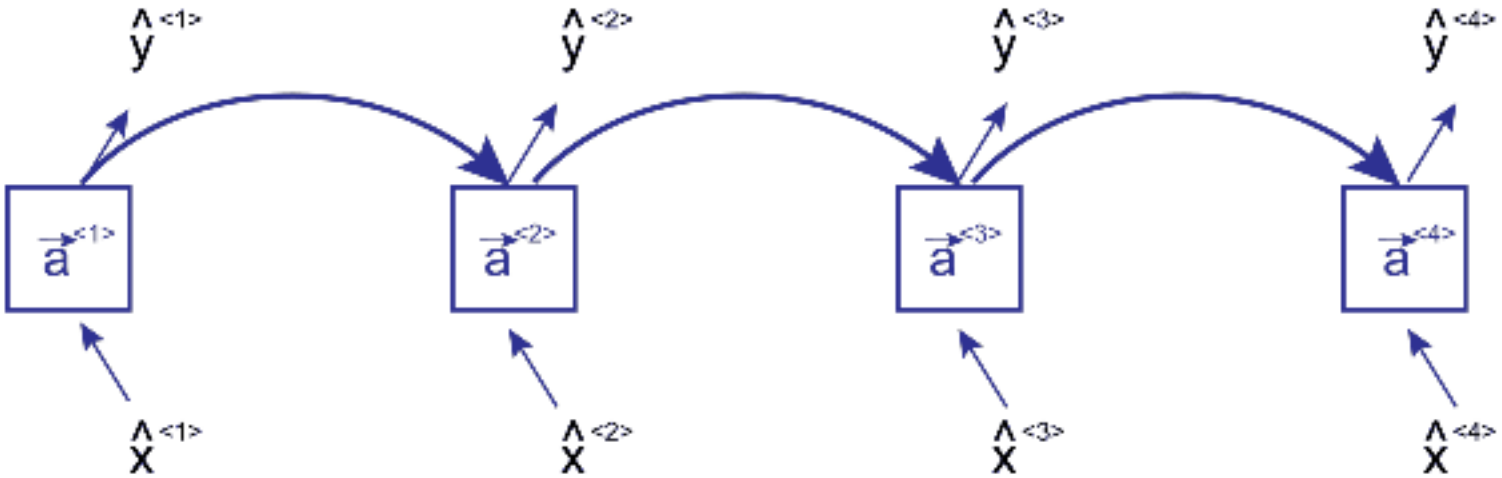
Bi-directional RNN



@raghavaggarwal0089

He said , "Teddy bears are on sale!"

@raghavaggarwal0089



Forward RNN (LSTM or GRU) network

@raghavaggarwal0089

Bidirectional layer

`Bidirectional` class

```
tf.keras.layers.Bidirectional(
    layer, merge_mode="concat", weights=None, backward_layer=None, **kwargs
)
```

Bidirectional wrapper for RNNs.

Arguments

- **layer**: `keras.layers.RNN` instance, such as `keras.layers.LSTM` or `keras.layers.GRU`. It could also be a `keras.layers.Layer` instance that meets the following criteria:
    1. Be a sequence-processing layer (accepts 3D+ inputs).
    2. Have a `go_backwards`, `return_sequences` and `return_state` attribute (with the same semantics as for the `RNN` class).
    3. Have an `input_spec` attribute.
    4. Implement serialization via `get_config()` and `from_config()`. Note that the recommended way to create new RNN layers is to write a custom RNN cell and use it with `keras.layers.RNN`, instead of subclassing `keras.layers.Layer` directly.
- **merge_mode**: Mode by which outputs of the forward and backward RNNs will be combined. One of {'sum', 'mul', 'concat', 'ave', None}. If None, the outputs will not be combined, they will be returned as a list. Default value is 'concat'.
- **backward_layer**: Optional `keras.layers.RNN`, or `keras.layers.Layer` instance to be used to handle backwards input processing. If `backward_layer` is not provided, the layer instance passed as the `layer` argument will be used to generate the backward layer automatically. Note that the provided `backward_layer` layer should have properties matching those of the `layer` argument, in particular it should have the same values for `stateful`, `return_states`, `return_sequence`, etc. In addition, `backward_layer` and `layer` should have different `go_backwards` argument values. A `ValueError` will be raised if these requirements are not met.

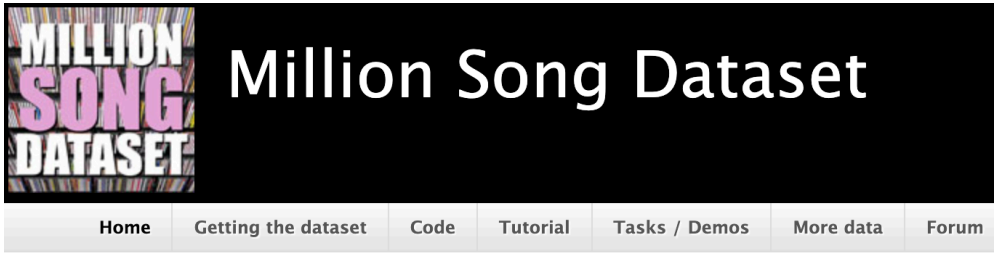
Audio Analysis

Why Should We Care About Audio Analysis?

Cultural Sociologist? How to predict the popularity of songs?

Ethnographer? How to process hundred-hours recordings?

Political sociologist? How to analyze politicians' speeches?



Let us see one example using audio data

What You Say and How You Say It Matters: Predicting Financial Risk Using Verbal and Vocal Cues

Yu Qin
School of Information
Renmin University of China
qinyu.gemini@gmail.com

Yi Yang *
HKUST Business School
Hong Kong University of Science and Technology
imyiyang@ust.hk

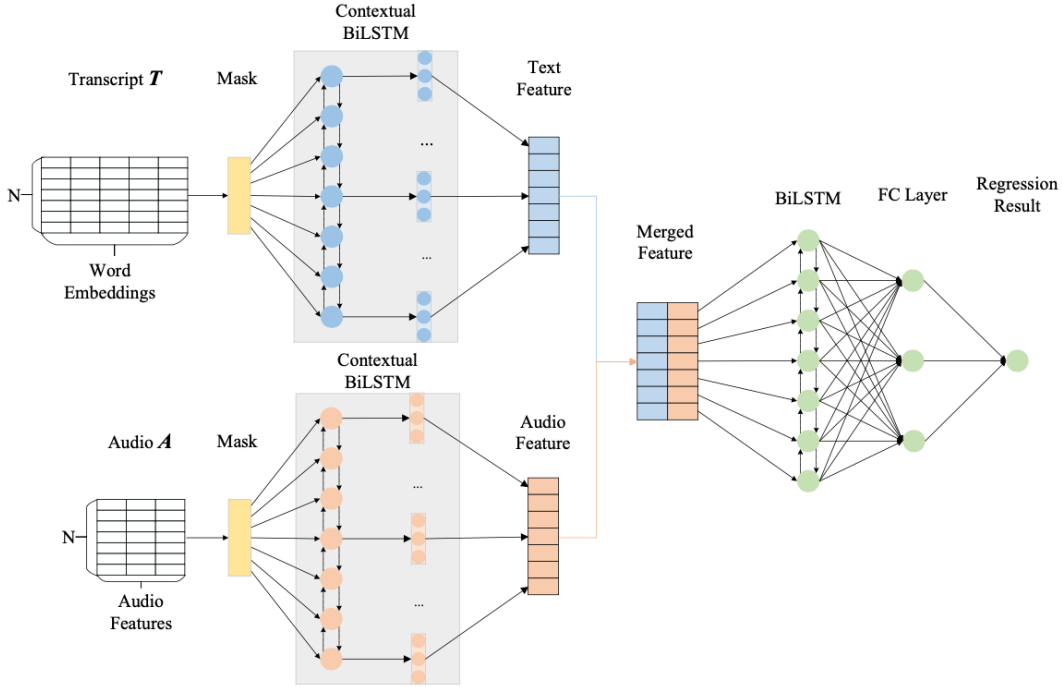


Figure 1: The proposed Multimodal Deep Regression Model (MDRM). The inputs to the model is a company's conference call audio file with correpsonding transcript. Each conference call consists of N sentences. The output variable is a numerical value, i.e., the company's stock price volatility following the conference call.

Thank you!

Yongjun Zhang, Ph.D
Assistant Professor
Dept of Sociology and
Institute for Advanced Computational Science
Stony Brook University
Yongjun.Zhang@stonybrook.edu
https:yongjunzhang.com