# Notes on Natural Language Processing with Transformers by Lewis Tunstall and Leandro von Wierra

compiled by D. Gueorguiev 3/30/24

## Introductory Notes

### The Encoder-Decoder Framework

Prior to transformers , recurrent architectures such as LSTM were popular in NLP. These architectures contain a feedback loop in the network connections that allow information to propagate from one step to another, making them good for modeling sequential data like text. An RNN receives an input (word or character), feeds it through the network and outputs a vector called *hidden state*. The model feeds some information back to its output through a feedback loop, which it can use on the next step. Unrolling the loop produces the “unrolled” network shown below: the RNN passes information about its state at each step to the next operation in the sequence – in a sense this simulates a memory attached to the network which collects condensed representation of the previous states which it uses in its sequential predictions.

State

RNN cell

Input

State

RNN cell

Input

State

RNN cell

Input

State

RNN cell

Input

State

RNN cell

Input

Figure: Unrolling an RNN in time

RNN cell

RNN cell

RNN cell

RNN cell

RNN cell

RNN cell

RNN cell

RNN cell

State

Transformers

aren’t

that

great

Transformers

sind

nicht

so toll

Figure: An encoder-decoder architecture with a pair of RNNs

Basics of Transformers

The Transformer architecture excels at

## Appendix

### Unreasonable Effectiveness of Recurrent Neural Networks

#### RNNs

What makes Recurrent Networks special?

Limitation of Feed Forward Networks and also Convolutional Networks:

that their topology is too constrained: They accept a fixed size vector as input (for example an image) and produce a fixed-sized vector as an output (for example probabilities of different classes). This kind of networks also perform their mapping using a fixed amount of computational steps because there are fixed number of layers in their topology.

RNNs do not have some of these constraints. The topology of the RNNs allow to operate over a *sequence of vectors* rather than just individual vectors.

A diagram of a number of rectangular objects

Description automatically generated with medium confidence ( 1 ) ( 2 ) ( 3 ) ( 4 ) ( 5 )

Figure: Each rectangle is a vector. Arrows represent functions (e.g. matrix multiply). Input vectors are shown in **red**. Vectors which hold the network state are shown in **green**. Output vectors are in **blue**.

( 1 ) denotes traditional processing in vanilla feed forward network which uses fixed-size input and has fixed-size output. Example: *image classification* with CNN. ( 2 ) sequence output topology – input has fixed size and output is of variable size i.e. *sequence output*. Example: *image captioning* which takes an image (fixed-size input) and outputs a sentence of words. ( 3 ) sequence input and fixed size output. Example: *sentiment analysis* where given sentence is classified as expressing positive or negative sentiment. ( 4 ) sequence input and sequence output. Example: *machine translation* where an RNN reads a sentence in English and outputs a sentence in French.

( 5 ) synced sequence input and output. Example: video classification where the goal is to label each video frame.

*Note*: in each of the relevant topologies there are no pre-specified constraints on the sequence lengths because the recurrent transformation can be applied as many times as we want.

## References

## [1] [Natural Language Processing with Transformers, Lewis Tunstall, Leandro von Werra, Thomas Wolf, 2022](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/natural-language-processing-with-transformers-revised-edition-book.pdf)

[2] [The Unreasonable Effectiveness of Recurrent Neural Networks, Andrej Karpathy blog, May 21, 2015](https://karpathy.github.io/2015/05/21/rnn-effectiveness/)

[3] [Long Short-Term Memory, Sepp Hochreiter et al., 1997](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/LongShortTermMemory.pdf)

[4] [LSTM Can Solve Hard Long Time Lag Problems, Sepp Hochreiter, Juergen Schmidthhuber, NIPS, 1996](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/NIPS-1996-lstm-can-solve-hard-long-time-lag-problems-Paper_Hochreiter.pdf)

[5] [Understanding LSTM: a tutorial into Long Short-Term Memory, R. Staudemeyer et al., 2019](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/TutorialOnLongShortTermMemory2019.pdf)

[6] [Sequence to Sequence Learning with Neural Networks, Sutskever et al, Google Research, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/SequencetoSequenceLearningwithNeuralNetworksSutsekver2014.pdf)

[7] [Turing Computability with Neural Nets, Hava T. Siegelmann, Eduardo Sontag, 1991](https://github.com/dimitarpg13/deep_learning_and_neural_networks/blob/main/literature/articles/computability/TuringComputabilityWithNeuralNets_Siegelman1991.pdf)

[8] [On the computational power of Neural Nets, Hava T. Siegelmann, 1992 (earlier version)](https://github.com/dimitarpg13/deep_learning_and_neural_networks/blob/main/literature/articles/computability/OnTheComputationalPowerOfNeuralNets_1992_Siegelmann.pdf)