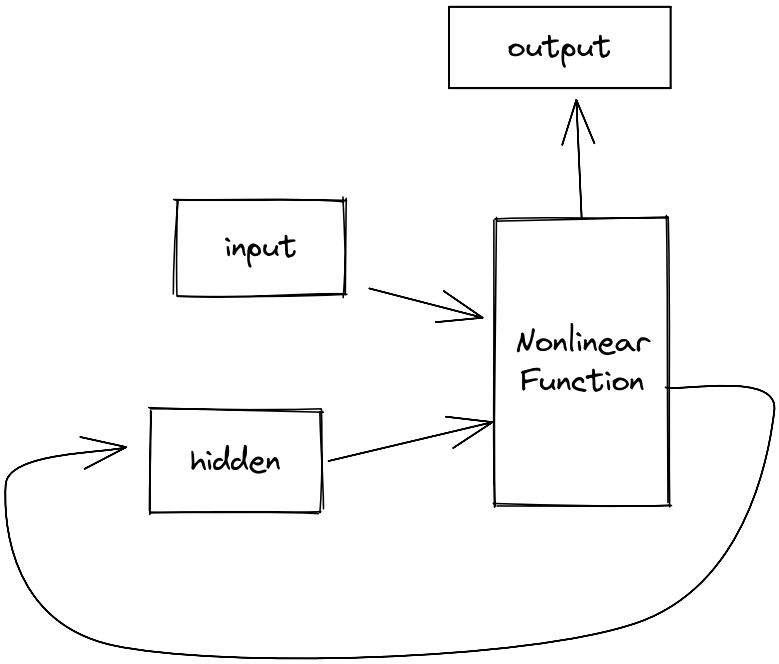
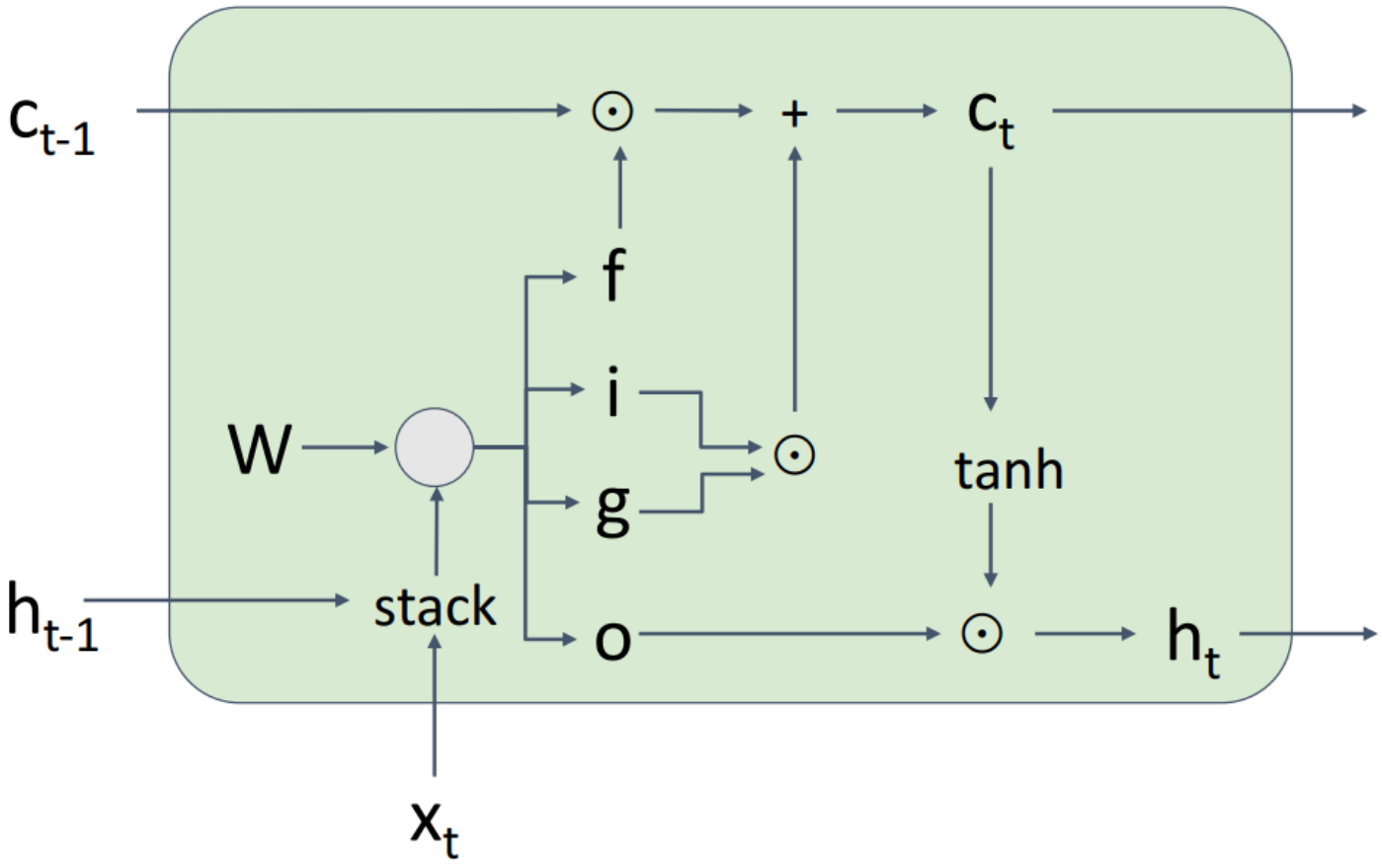
RNN

Pretty basic concept. Inputs fed back into network. Layer normalization helps avoid extremes & can get longer recurrences. (Suffers from processing large amounts of text, since gradients don't flow well.) Basic functionality:



LSTM (long short term memory) can fix gradient flows since there's a direct path for the gradient (kind of like residual connections).

Long Short Term Memory (LSTM)



Attention

Query vs. Keys vs. Values

Query = what do you want to know?
Keys = labels for the values (to be matched with the queries)
Values = values.

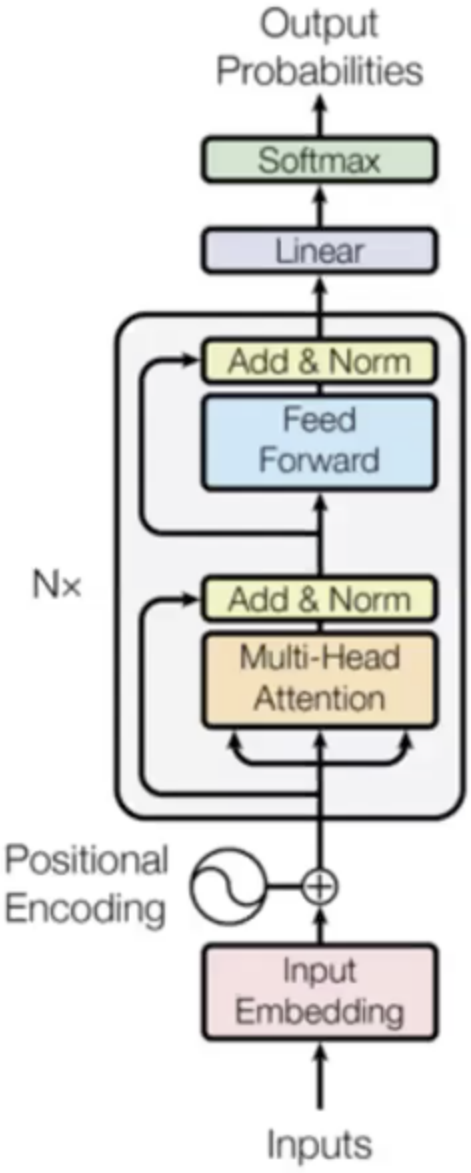
How it works: Query * Keys (dot product) tells how well each label works for your query. After a softmax layer, take the sum of this with the values.

Self-Attention

Queries, keys, and values are all from the same source, but just different projections. I.e., if they started out as R^N, you might get R^k, R^k, and R^v outputs.

DL for NLP 1 & 2

Started with fixed size, then RNNs, now transformers. Good picture of model with transformer:



Tricks to use:
Warm up, layer normalization, label smoothing (i.e. spread out the probability so everything isn't absolute).

Reasons to use transformers:

- very parallel
- very good at scaling up
- direct connection between all words = very good at context
  Issues:
- Grows much more costly (quadratically) as the size of context increases.

Decoding Language Models

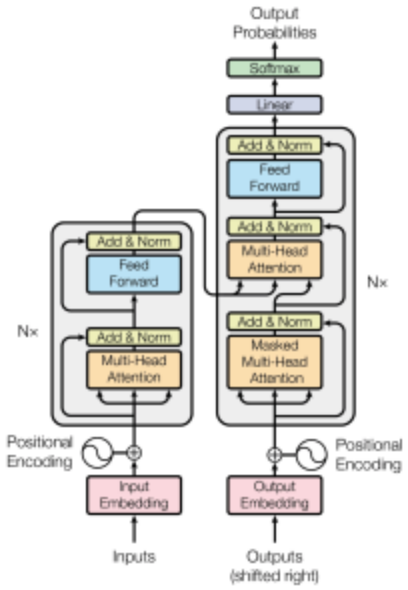
Random English:
Beam search works fairly well, sometimes nonsense, more often very short, generic phrases (e.g. "yes", "no", "thanks!"). Top-k (randomly sample from one of the top-k choices) works much better.

Translating/other tasks: Use sequence-to-sequence transformers!

Encoder Stack

Each input word attends to:

- All other input words



Decoder Stack

Each input word attends to:

- all input words
- All *previous* output words

How to train: Several methods, such as back-translation (convert horse to zebra back to horse).

You can also actually train on each language separately, by having some internal "language" to translate to, and then back to whichever language you want as output. This makes it so you can use massive amounts of each text without the appropriate translation. I think this is what Google Translate now does.

Unsupervised learning

word2vec
GPT
ELMo -- two training models, one left->right, one right->left.
BERT -- like word2vec but with transformers

Often good to pretrain on one of above and then finetune on specific tasks.

Open questions

How do we do long documents?
Can we be more efficient in fine tuning?

Other Notes:
TO BE DONE LATER