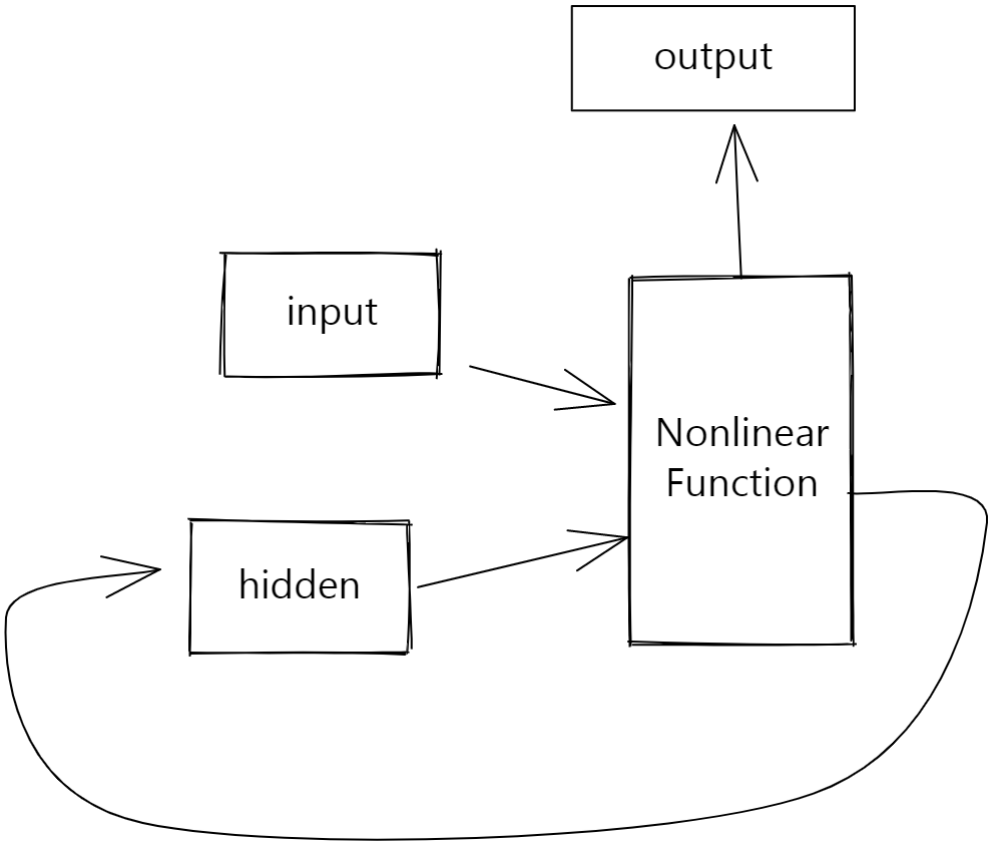
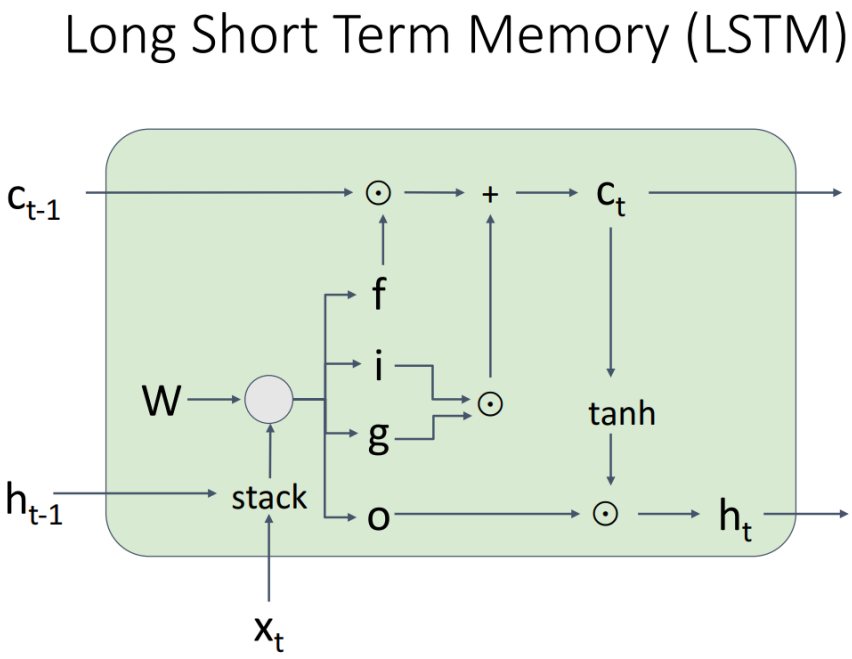
Lecture Notes

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).



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.

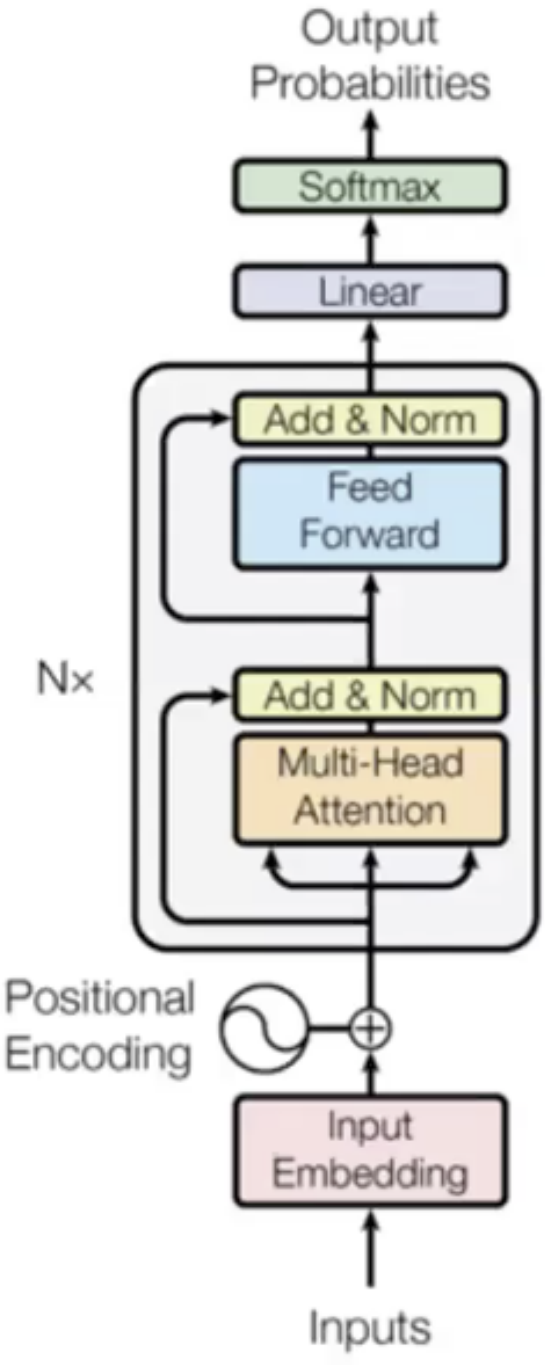
How it works: Query * Keys (dot product) tells how well each label works for your query. After a softmax layer, take the sum of this with the values.

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:


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs

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

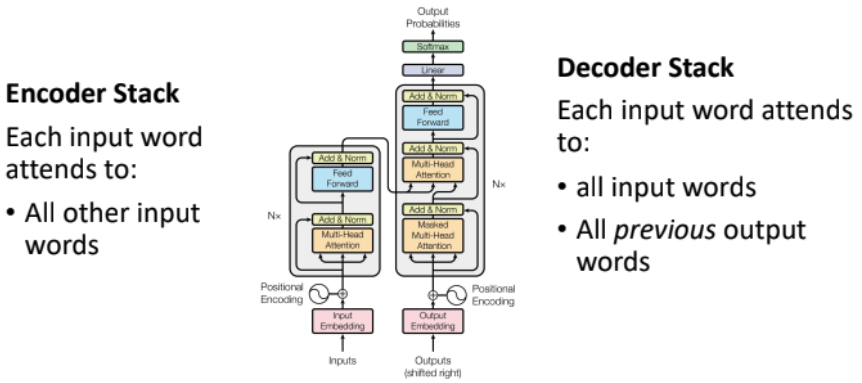
Reasons to use transformers:

- 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!



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?

Reinforcement Learning

Basic idea: Train an agent to maximize a cumulative reward.
Difficulties: Sampling reward is typically costly, so we need to be sample efficiency. Often, the reward function is implicit (e.g. in chess) and you need to figure it out from the rules.

Sometimes, reward depends on where you are in space--you need to explore to make accurate predictions.

Q-learning: Use a neural network to approximate the reward function. Then, take whatever action maximizes your score according to the Q-function.
Policy Gradients: Train neural network to get probabilities you should take each action. Use REINFORCE to compute gradients.

Imitation learning: Get data from human experts, try to imitate them. (Supervised.)

Leveson Lecture

What is STAMP?

Models of loss

chain of event -- linear, one thing leads to the next directly.

Defense in depth -- several backup systems. E.g. in the Bhopal chemical, they had storage in nitrogen gas (inert), regular inspection, double walled barrels, etc.

Exercise:

Causal factors: Too many false positives on safety issuesv led the operators to think real problems were just more false positives.
Root cause: "Safety" features were restrictive in ways that didn't actually help. They were designed to appear good, but weren't actually useful to the operators.
Questions: How can we ensure that "safety" features are actually doing more good than harm/stress? In driving, for example, is forcing people to do head checks every time they change lanes or turn really safer, or is this arbitrary 100% rule just a way to make lawmakers look like they're doing something to make driving safer?

Back to Chain of event model

Root cause is arbitrary. Making one up provides "illusion of control". Problem with this is that it often leads to treating symptoms, not the actual problem. Chain of event is not enough for a causal model.

Level 2 conditions

What causes the events?
Most of the safety features were poorly designed in the Bhopal accident, which is one cause of why they failed.

Level 3 systemic problems?

Similarly to how a lot of companies higher cybersecurity "experts" to look like they're handling data securely, Bhopal made "safety" features to look like they were handling MIC safely. In reality, however, the safety features did little to guarantee that the system was actually safe.

Pilots:

Almost always blamed, even though there's lots of confounding factors. Often blamed, even when confounding factors clearly are a big reason, because other pilots *don't* get in crashes from those same problems.

Complexity

Sometimes, things go wrong not because of mechanical failures, but because people don't understand them.

E.g. Mars Polar Lander -- crashed because of software requirements not matching what the software developers were told. Noise -- unexpected by software developers -- caused it to cut off its engines.

Warsaw A320 -- software incorrectly thought the plane hadn't landed.

Complexity is a COMPONENT INTERACTION ACCIDENT.

Software

Infinite complexity possible. Context determines safety. (Software never fails. Only how it is implemented, or what situations it is put into causes something else to fail.) They have design errors, not mechanical failures.

STAMP

We need to integrate things. Then she spends 30 minutes talking about how its a "paradigm change" and "super needed", but doesn't actually says what it is.

She gets sidetracked about this "analytic decomposition". Okay, that seems pretty obvious... Aparently, though, that's still "not the problem".

System Theory

Focuses on systmes "taken as a whole". "Emergent properties". Okay, so how is this actually **USED**? Lol, emergent properties. Like, what does that even mean? It's just another thought stopper. I don't think she intends to actually impart useful information beyond a few case studies of system failures.

Aha, we need to somehow "control" the emergent properties. Explain how your control doesn't become a part of the system, in which new emergent properties will arise.

So, the big paradigm shift is viewing safety as a "control" instead of "reliability" problem? What does that even mean? If you could control failures, obviously you would just stop making them. Redundancy sounds more like "reliability" than control. Fail-safe design is laudable, but isn't the whole point about your spiel on "controlling" failures figuring out how to recover from unexpected failures??

Aha! Their new way of modeling is making everything an input-output diagram. Inputs = arguments passed to software/things observed by human. Outputs = decisions. For some reason, her model has two separate blocks for "process model" and "decision making". Maybe to point out two places where failures can occur?

STAMP again??

System-Theoretic Accident Model and Processes
Still don't say what it actualy IS or HOW TO USE IT. Just that it somehow incorporates more factors into the "blame".

Circular causes:

Root causes are not individual nodes. They are forces emerging from bad feedback loops.

Example: Columbia shuttle loss -- one circular cause is performance pressure leads to higher expectations, leads to higher performance pressure.

ROI on using STAMP:

ONE industry contractor **claims** they are seeing 15-20% return on investment when using StPA.