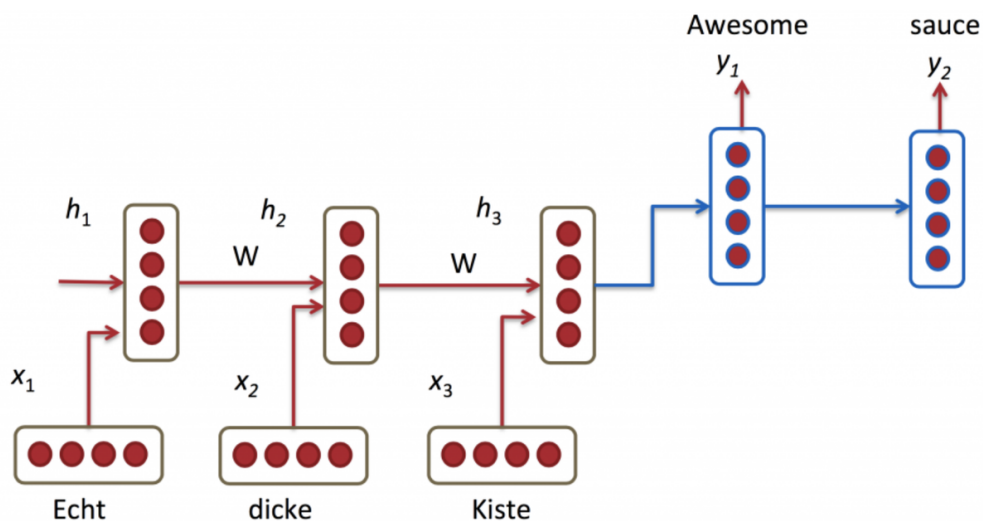


# neural machine translation

- MT takes a lot of parallel data → the bible is one such, translations to lots of languages are available for a ton of text
- traditional MT systems were complex and required lots of teams to build them. they had different parts like word alignment, word re-ordering based on how words in a language are usually laid out and many other custom models for subproblems.
- these systems involved a search module to search for possible translations as well as a language model to say “is this sequence of words likely?”
- it required big teams and lots of complex feature engineering. enter deep learning → simple, easy to implement and can be done with a couple of people’s couple of months to get state of the art results
- one way to do a MT with RNNs is to use an encoder decoder architecture. the encoder reads in a sentence/phrase and builds a “context vector”, also called a sentence embedding, which is the vector output by the last step of the encoder. the context (or sentence embedding) then gets decoded by a decoder. the decoder RNN outputs a special character to indicate that its done decoding.



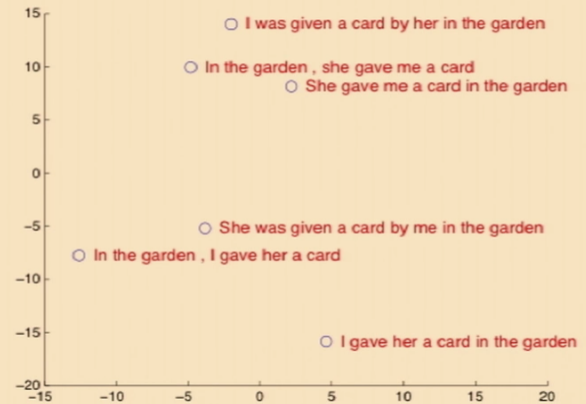
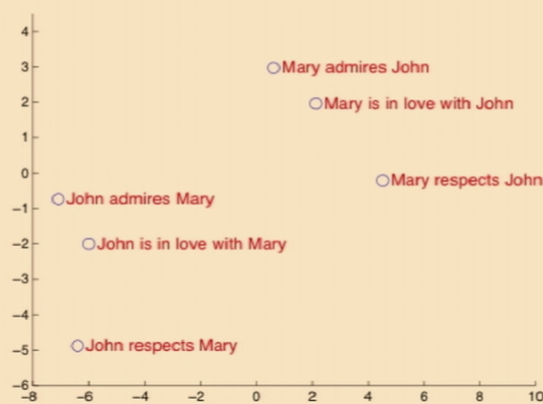
- Sometimes it helps to feed the context vector to every step of the decoder RNN and not just the first.
- It might also help to use bidirectional RNNs
- another trick for some language pairs is to learn to map the reverse of the input to the output instead of input to output. If the language pair is such that if a word occurs early in one language and late in another, then it can help to shorten the length of “remembering” which helps to avoid long term dependency problems like vanishing gradient etc.
- another trick that has worked is to feed in the input twice and then teach the model to map that to the translated sentence. maybe this helps the model remember stuff again before producing the final sentence embedding.
- RNN units to help us better model long term dependencies. GRUs and LSTMs

## GRUs

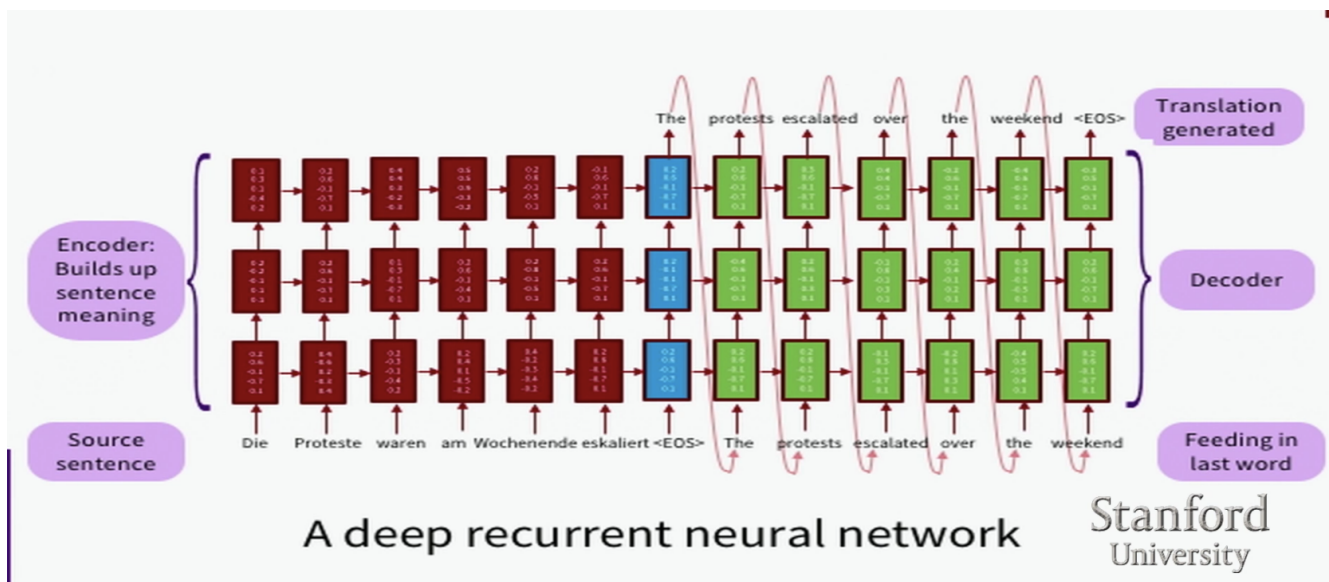
- Update gate  $z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right)$
- Reset gate  $r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right)$
- New memory content:  $\tilde{h}_t = \tanh \left( W x_t + r_t \circ U h_{t-1} \right)$   
If reset gate unit is  $\sim 0$ , then this ignores previous memory and only stores the new word information
- Final memory at time step combines current and previous time steps:  $h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$

- see notes on LSTM and GRU from deep learning book notes.
- with sigmoidal units that show up in RNNs, you have to think about initialization. if you initialize to the extremes, the gradients are very small here and it won't make much progress during training. so don't initialize at the extremes.
- This PCA or t-SNE projection of context vector shows that syntactically different sentences with the same semantics are being projected to nearby vectors. That's pretty amazing!


## PCA of vectors from last time step hidden layer



- we want models to be invariant to simple syntactic changes when the semantics are kept the same
- MT really can help to scale businesses across borders, enable the web to become accessible to people across languages, do the same for books potentially and enable people from different languages to communicate with each other.
- RNNs can be deep with many hidden layers, each being recurrent.



- Because there is a single context vector that needs to produce all the outputs, you are putting pressure on the forget gates not doing too much forgetting.
- Also note that the context vector is only fed into the first step of the decoder. that step needs to pass on relevant information to further time steps so as to preserve information until the last time step has generated its word
- another approach is to feed the context vector into every time step of the decoder.
- Why do neural systems perform so well at MT?

- 
- these systems are all built to translate from a given language to another language, trained for each language pair. what if we can share some of these components? some ways to do it:

- this data sharing improves the quality of translation for languages where not too much training data is available.
- with no data sharing you'd have to support 80 languages, you'd have to train 6400 models! well what they actually used to do was translate to intermediate languages and then translate to target language. but this sentence embedding in neural machine translation is the new intermediate language.
- the problem with vanilla seq2seq for NMT is that for long sentences, long term dependencies becomes a real problem. enter attention.
- don't pay attention to everything all the time!

- 
- ideally, we'd like to look at what we've generated so far to figure out what to pay attention to
- 

- the attention weights are normalized to sum to 1 using softmax, which gives the prior of "limited attention, decide what you want to focus on". → this softmax is not shown above. softmax attention is differentiable and allows us to learn weights to determine where to focus on for what inputs.
- remember, the decoder outputs words by outputting a probability distribution over output words. instead of simply picking the next word of highest probability which is greedy and can be incorrect, we can do a [beam search](#) over the next 5-10 time steps and keep the top  $k$  candidates at each time. a theoretical person would say you have to look all the way and pick the global optimum but of course this is infeasible → think about your vocab size.
- how to evaluate machine translation quality? Problem is there is no simple single number that gives good learning signal. Probably shouldn't only count 1 output as correct. Also want a way to express that its partially correct. some techniques:

- We can also cross reference with many reference translations and allow us to match with any of them

# Stanford University



- [illegible]

- another strategy is to use some heuristics to figure out only words that can be there in the translation and include that in the set of output possibilities.