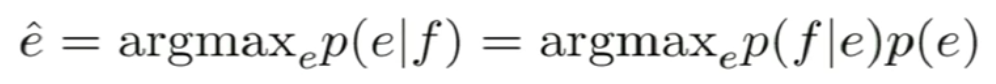
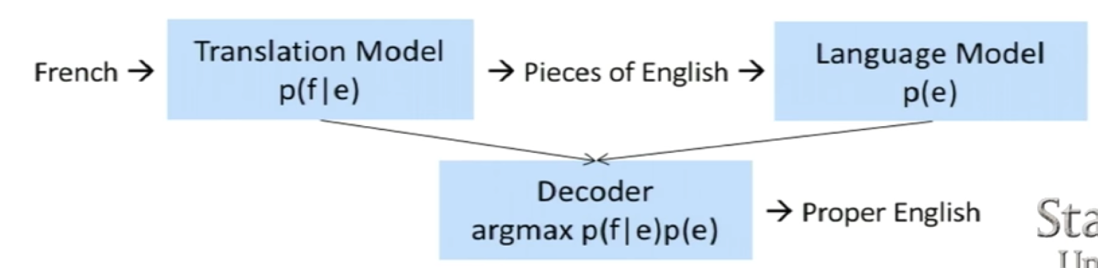
Lecture 9 | Machine Translation and Advanced Recurrent LSTMs and GRUs

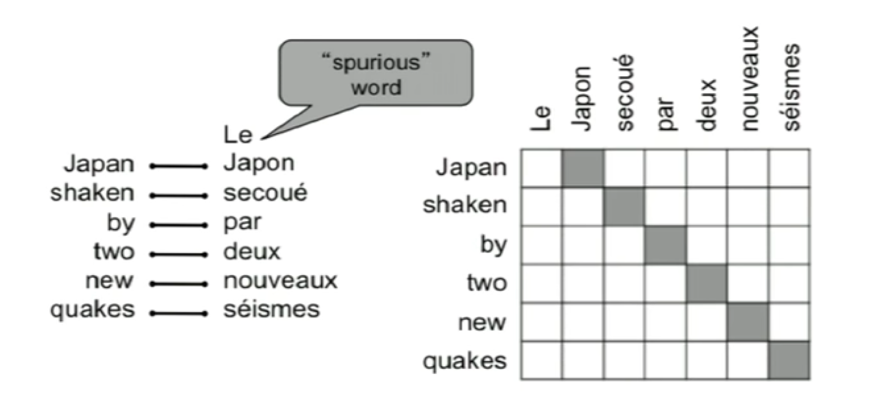
* LSTMs (1997) and GRUs (2014)
* All machine translation uses statistical methods on parallel corpora, where you have a corpus in one language and you have another corpus which it’s a translation of the first corpus in another language
* Current statistical machine translation systems (description)
  + Source language, f (e.g. French)
  + Target language, e (e.g. English)
  + Probabilistic formulation using Bayes rule:



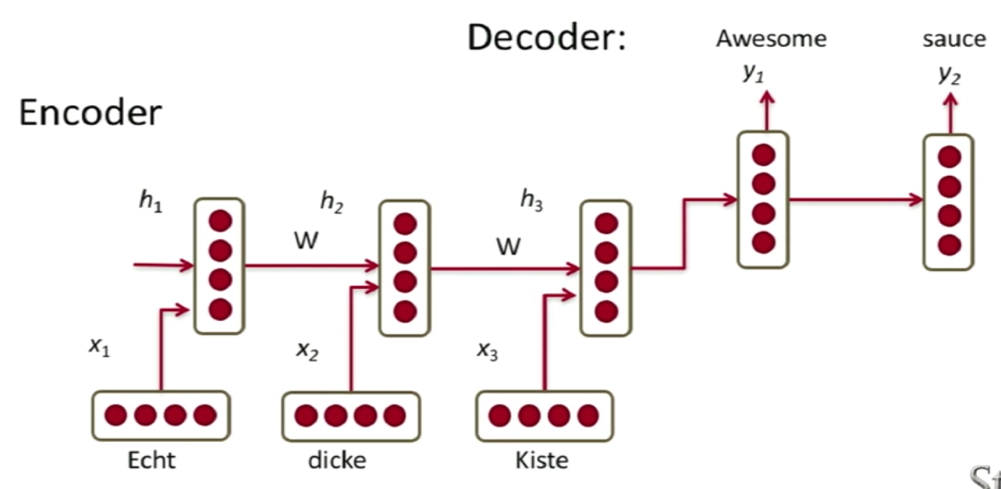
* + Translation model p(f|e) is trained on parallel corpus
  + Language model p(e) is trained on English only corpus



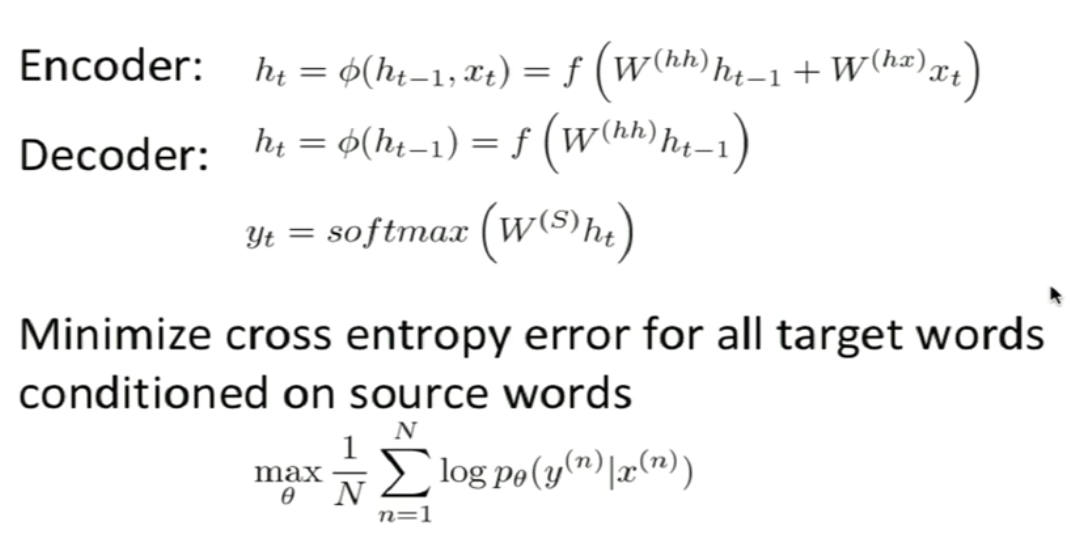
* + How to train this model?
    - Alignment
      * Know which word or phrases in a source language would translate to what words or phrases in target language (difficult)



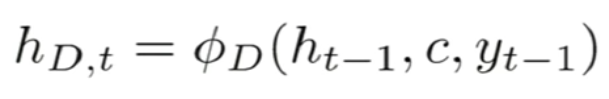
* + - * We could face ‘zero fertility’ problem, which is word that is not translated
      * We also have situations where we have one-to-many/many-to-one/many-to-many alignment e.g. implemented 🡪 mis en application
    - You have to also consider reordering of translated phrases
    - Decoding
* Deep learning basic machine translation

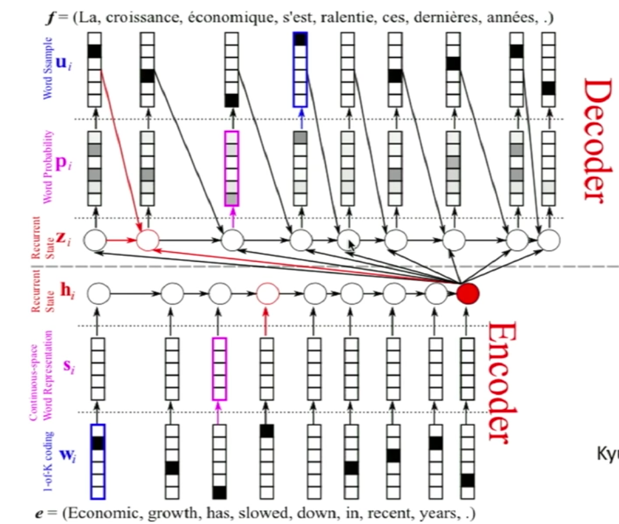


* + The last vector of the encoding component of the model has to capture the entire phrase! But usually only 5-6 words can be captured and after that, we couldn’t really memorise the entire context of the sentence before

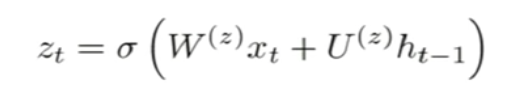


* RNN translation model extensions
  + Train different RNN weights for encoding and decoding
  + Compute every hidden state in decoder from
    - Previous hidden state
    - Last hidden vector of encoder, c = h\_t
    - Previous predicted output word, y(t-1)

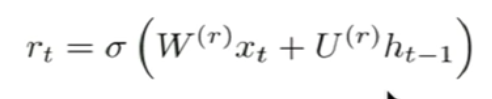




* + Train stacked/deep RNNs with multiple layers
  + Potentially train bidirectional encoder or train input sequence in reverse order for simple optimisation problem
* Gated Recurrent Units (GRUs)
  + The main idea is to keep around memories to capture long distance dependencies, allowing error messages to flow at different strengths depending on the inputs
  + GRU first computes an **update gate** based on current input word vector and hidden state. Update gate controls how much of past state should matter now. If z close to 1, then we can copy information in that unit through many time steps, therefore, less vanishing gradient



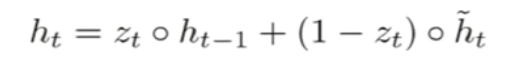
* + GRU computes **reset gate** similarly but with different weights



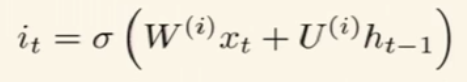
* + New memory content:



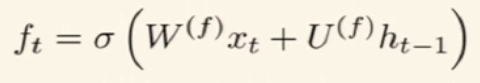
* + If reset gate unit is around 0, then it ignores previous memory and only stores the new word information, allowing model to drop information that is irrelevant for future output
  + Final memory at time step combines current and previous time steps:



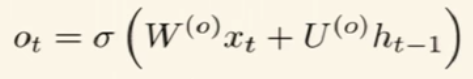
* + Units with short-term dependencies often have very active reset gates
* LSTMs
  + Allow each time step to modify
    - Input gate (current cell matters) – how much we care about the current vector



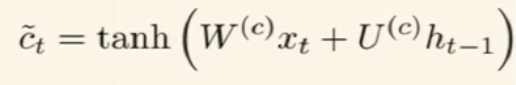
* + - Forget (gate 0, forget past) – Should I forget the past?



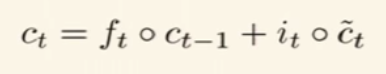
* + - Output (how much cell is exposed)



* + - New memory cell



* + Final memory cell



* + Final hidden state

