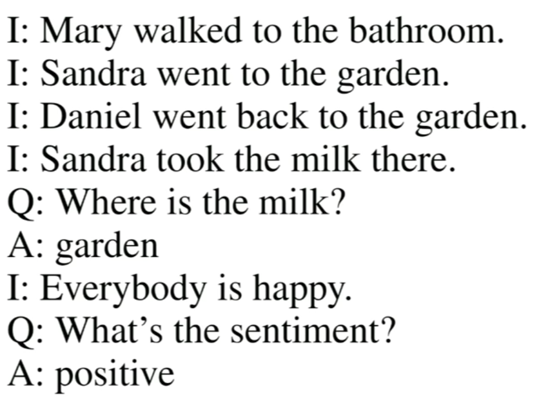
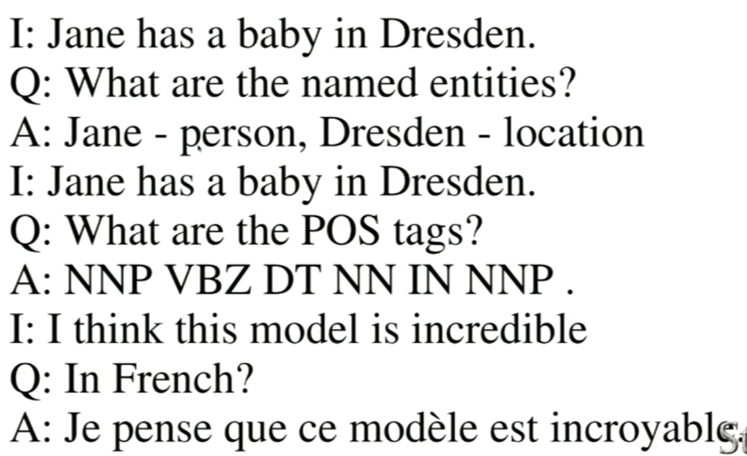
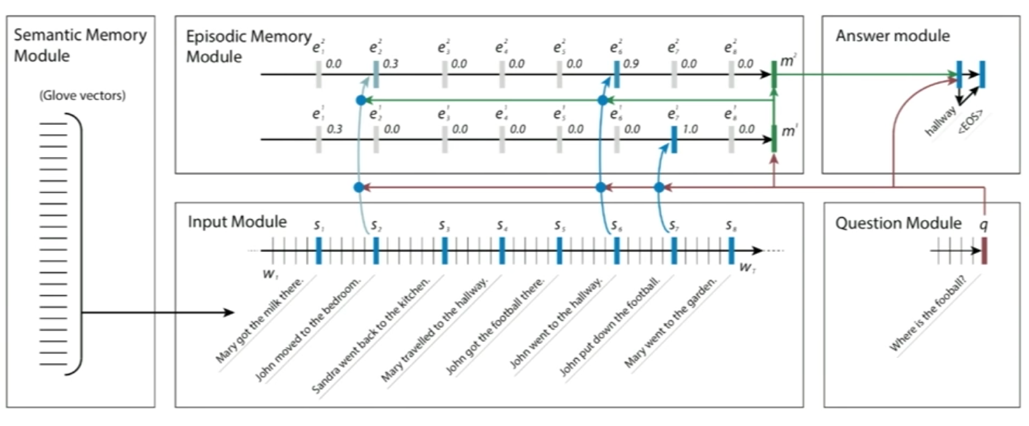
Lecture 16 | Dynamic Neural Networks for Question Answering (QA)

* Can all NLP tasks be seen as question answering problems?
  + QA examples:

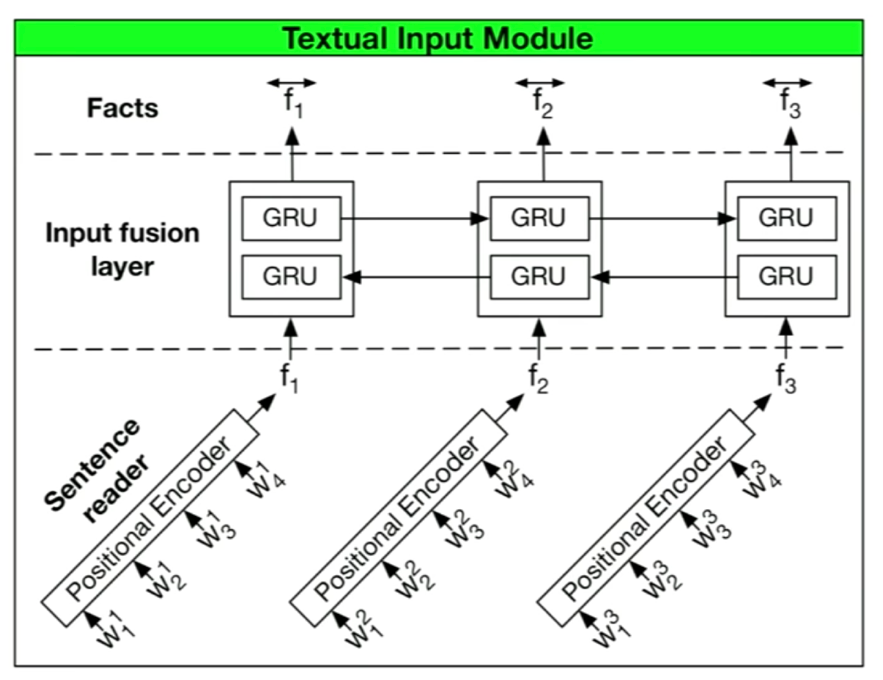




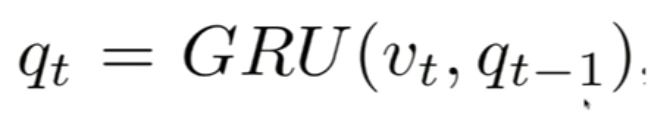
* A joint model for general QA
  + First major obstacle is for NLP, no single model architecture with consistent state of the art results across tasks
    - Question answering, state of the art model is strongly supervised MemNN
    - Sentiment Analysis, state of the art model is Tree-LSTMs
    - POS tagging, state of the are model is Bi-directional LSTM-CRF
  + Second major obstacle is that fully join multitask learning is hard:
    - Usually restricted to lower layers
    - Usually helps only if tasks are related
    - Often hurts performance if tasks are not related
* Dynamic Memory Networks (only tackle the first obstacle)



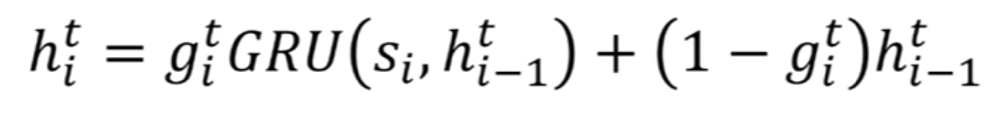
* + **Semantic Memory module:** Starts with simple word vectors like glove or word2vec
  + **Input module:** We have a recurring neural sequence, such as GRUs, that goes over the input and computes the hidden state at every word and at every sentence
  + **Question module:** GRU for the questions and in fact sometimes you can share the weights between the question and input modules. Basically, use a GRU to compute a question vector. That question vector, q, is going to be the last hidden state of the GRU after it went for every word of the question
  + The question triggers an attention mechanism that goes over all the hidden states of all my inputs and look for facts that seems relevant for the question at hand. Whenever there is a strong attention being paid to a specific sentence, we are going to give that sentence as input into another GRU in the **Episodic Memory module**
  + Therefore, the first GRU within the Episodic Memory module agglomerates only the facts that are pertinent or relevant to the question at hand. Essentially it is a **filtering GRU** that tries to only keep track of what’s relevant for the current question. Now we will define this memory state, m\_1, as the last hidden state of the GRU
  + For the next iteration, “John put down the football” but we don’t know where John is so we have now stored in vector m\_1 that in order to answer the current question, it seems pertinent to know where John is. Therefore, as we go over the input again, we will take **m\_1 vector** and **q vector.** Therefore, we now pay attention to every fact that mentions both John and/or football, which the attention scores are stored in m\_2 vector
  + m\_2 vector is now given as input and so the zero-time step in **Answer module** is yet another GRU that has our standard soft max classifier at every hidden state to produce an answer
* Input module
  + Standard GRU. The last hidden state of each sentence is accessible
  + One improvement that we have made on the second iteration is we have a Bidirectional-GRU instead of a unidirectional GRU



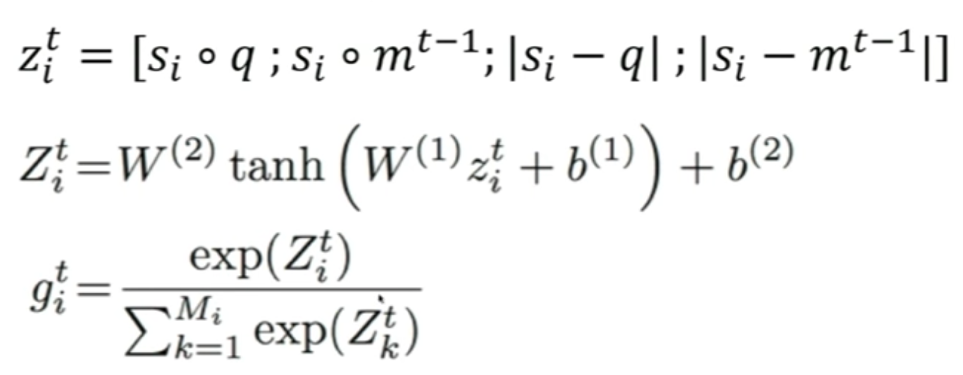
* + Each sentence is represented via the concatenation of both the left to right and right to left direction
* Question module
  + A standard GRU



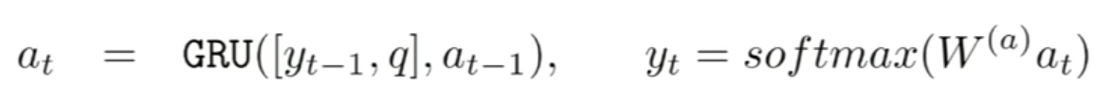
* + v\_t is our word vectors
* Episodic memory module



* + We have attention mechanism here captured by g\_t (a single scalar number) where t is the tth time that we went over the entire input
  + Main idea is that we have a global gate on top of a standard GRU and so this global gate will basically say whether a particular sentence matter or not
  + Therefore, if g\_i\_t, at the ith sentence and tth iteration, is 0, then we will basically just entirely copy the previous h\_vector and moved it forward
  + Last hidden state, m\_t
  + **How do we compute g?**
    - Gates are activated if sentence is relevant to the question or memory
    - When the end of the input is reached, the relevant facts are summarised in another GRU

****

* + If summary is insufficient to answer the question, repeat sequence over input
* Answer module



* + GRU with one little trick, which is we don’t just have the previous hidden state but we also concatenate the question vector at every single input state as well as the previous words output
* Comparing to MemNets
  + Similarities:
    - Both have input, scoring, attention and response mechanisms
  + Differences:
    - For input representations, MemNets use bag of word, nonlinear or linear embeddings that explicitly encode position
    - MemNets iteratively run functions for attention and response
  + DMNs show that neural sequence models can be sued for input representation, attention and response mechanisms (naturally captures position and temporality)