Deep Learning and Natural Language Processing - Applications

Vincenzo Scotti,
Ph.D. student
vincenzo.scotti@polimi.it

arcslab
adaptable, relational and cognitive software environments

N.L.P. – A.Y. 20-21

Contents

- Basic applications
  - Text classification
  - Fine-tuning transformers
  - Audio classification
- Text analysis
  - POS tagging and NER
  - Parsing
  - Neural machine translation
  - Question answering
  - Chatbots
    - IR
    - Generative
- Voice analysis
  - ASR
  - TTS
  - Speaker diarization
  - Noise cancelling
- Multimodal analysis
  - Emotion recognition

All the slides here are to be considered as informative

BASIC APPLICATIONS

Overview

- Simplest applications possible in NLP include the training of a classifier
- Inputs, either speech or text are treated as time series of features (2D tensors or 1D feature maps)
- We distinguish between 2 tasks in this sense
  - **Classification**: when you have to associate a sigle class to the input sequence
  - **Continuous labelling**: when you have to associate a label to each element (or each subset of elements) of a single input sequence
- Other problems involve the training of generative models, to have a synthetic text or speech as output
  - Similar to regression problems

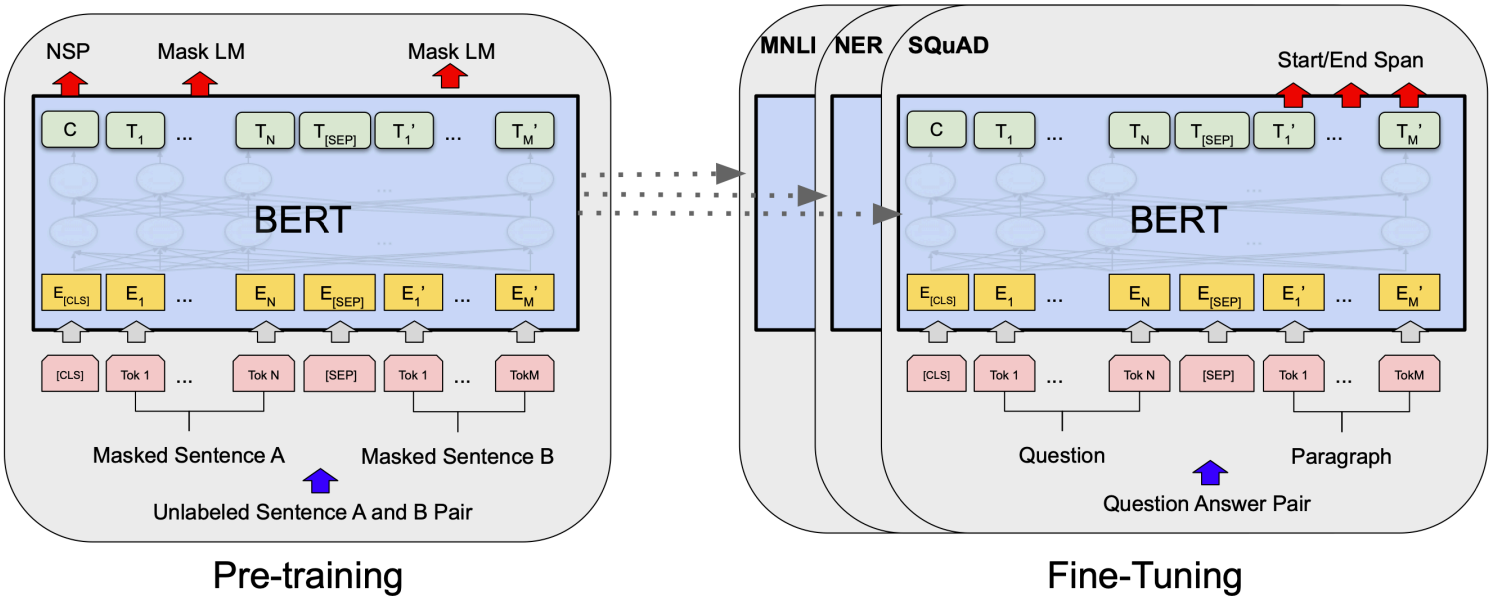
Text classification

- Usually the input $\mathbf{X}$ in these cases is given as a sequence of word embeddings
- Any kind of layer can be used for this input
  - 1D convolutions and pooling
  - (Bidirectional) Recurrent
  - Attention
- The difference is on how the last layers are treated
  - It depends on the problem
- The aforementioned layers produce a continuous output, directly compatible with continuous labelling
  - Just add a fully connected layer sliding over input, or use either a 1D CNN or a (Bi)RNN all with $\mathrm{softmax}()$ activation
- To deal with classification is necessary to flatten information along the time axis
  - It's possible to do so with GAP, MAP, taking the last state of the RNN or having an attention layer with a single query vector (also by fixing input length and unrolling the feature map, but not so elegant and transferable)
  - Finally put a fully connected layer with $\mathrm{softmax}()$ activation on top of it

Fine-tuning transformers

- Most solution for text analysis nowadays rely on transformer and their strong representation capabilities
- The idea is to use them for any king of task
  - Start from unsupervised pre-training on huge unlabelled data set
  - Refine over a specific task
  - They work for both generative and discriminative problems
  - A part from the input tokens extracted from text, they use some special tokens to produce their outputs
- In general (but it is not necessarily so) BERT-like models are fine tuned for discriminative tasks and GPT-like for generative tasks

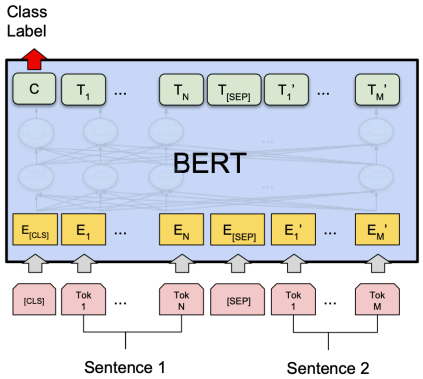
Fine-tuning transformers



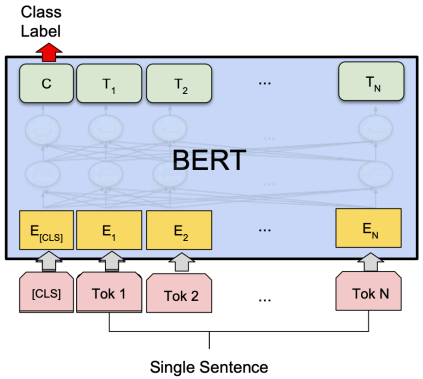
Fine-tuning BERT

- BERT introduces 2 special tokens to its input to perform its tasks
  - '[CLS]' for classification tasks
  - '[SEP]' to mark a split over the input sequence
- It can be fine tuned for many tasks
  - Text classification (use '[CLS]')
  - Text pair classification (use '[CLS]' and '[SEP]')
  - Sequence labelling (simply add sliding fully connected layer on top)
  - But also question answering (give as input text reference and question followed by '[SEP]' and use the same mask from training over the input where the answer should be, in that position it will output the answer)

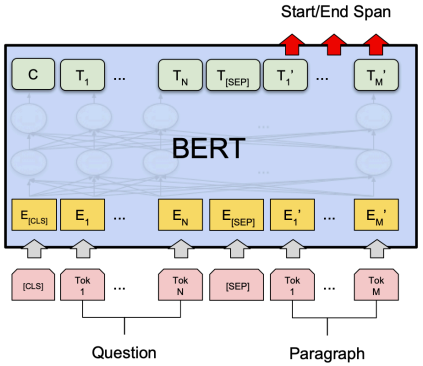
Fine-tuning BERT



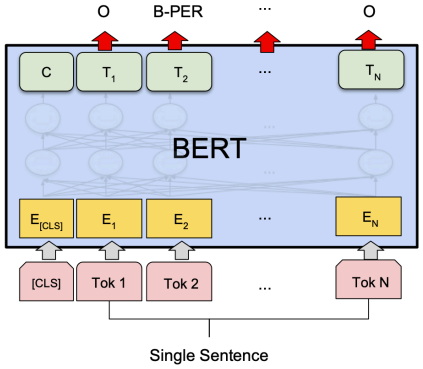
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1

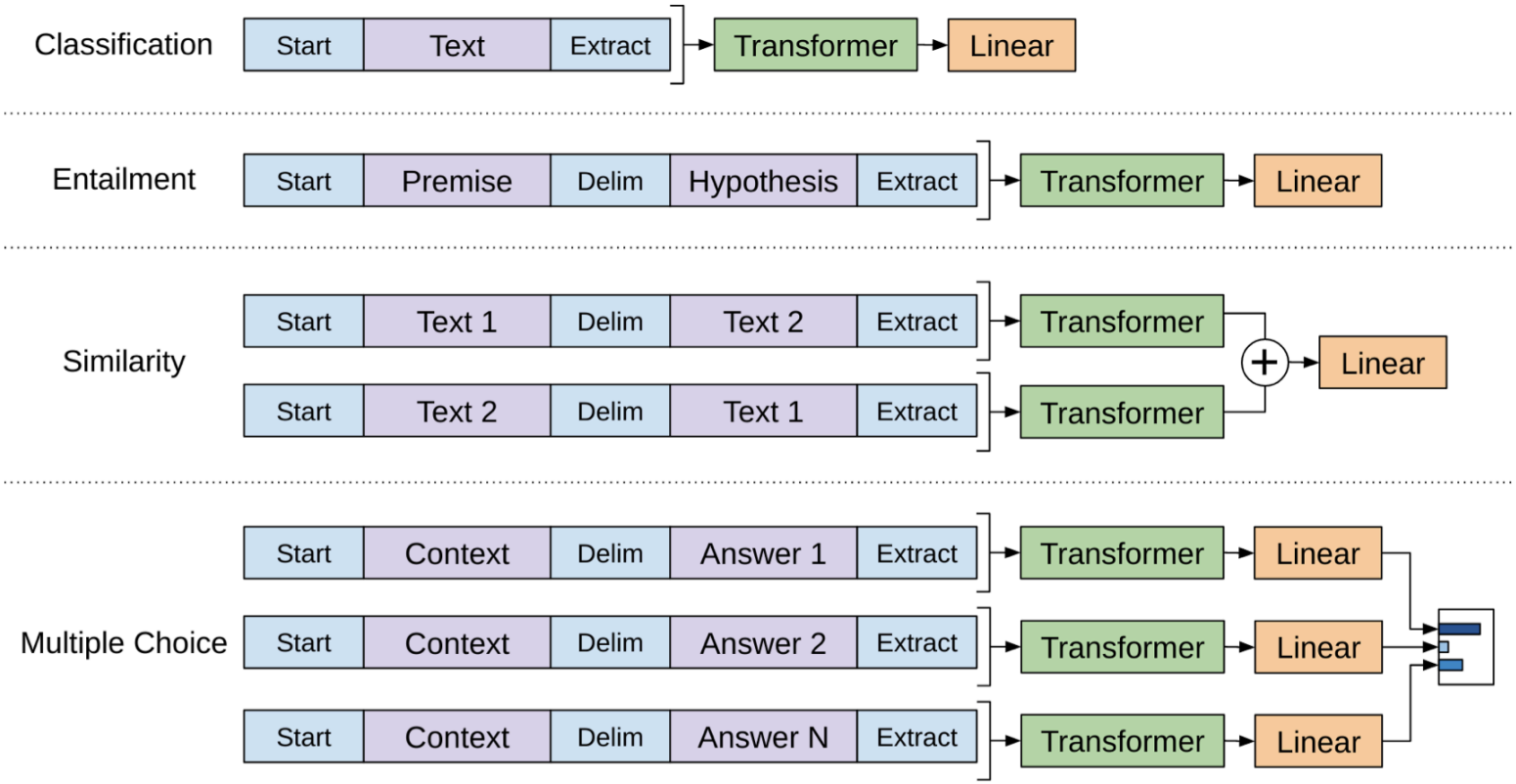


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Fine-tuning GPT

- GPT can be used for text encoding in the same way of BERT
  - And performs the same classification/labelling tasks in a similar way
  - It use three main tokens:
    - ‘<start>’
    - ‘<delimiter>’ (similar to ‘[SEP]’)
    - ‘<extract>’ (similar to ‘[CLS]’)
- Its main capability, however, is in text generation
  - In particular from the second version it it trained to perform multiple tasks at one by introducing special delimiter tokens that identify the commands
- Without fine-tuning, it is already capable of:
  - Summarizing (give as input original text and ‘<summarize>’ delimiter, then start decoding output to have summary)
  - Translation (give as input original English text and ‘<translate en to fr>’ delimiter, then start decoding output to have the French translation)
  - Question answering…

Fine-tuning GPT

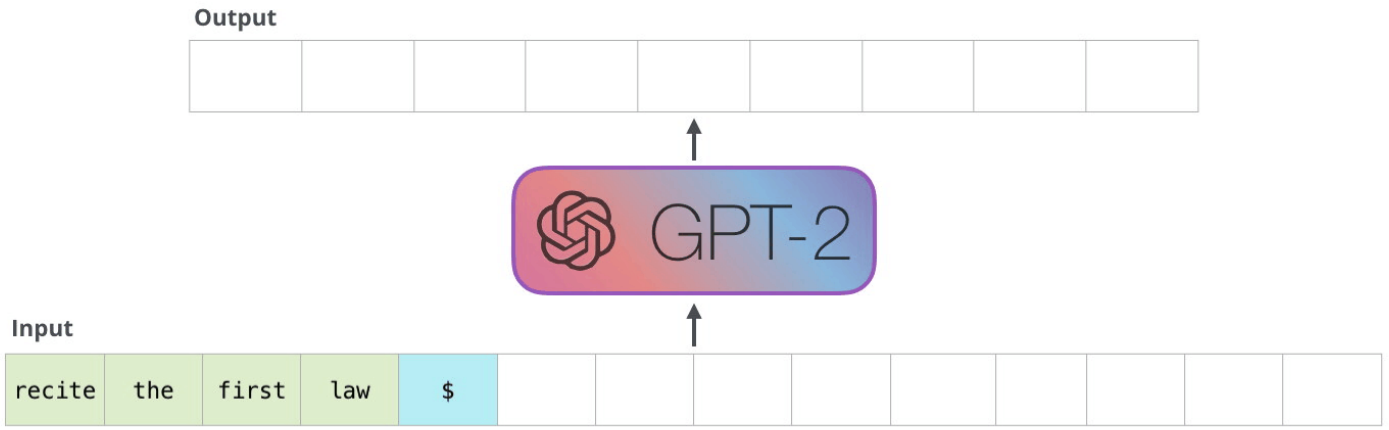


About GPT-2 and GPT-3

- From the second version of GPT the idea is to train a multitask learner
  - They aimed at a training so generic that the output model could perform many tasks without fine tuning (they appear naturally in text)
- From GPT-3 three options
  - Zero-shot task transfer: directly produce output for a task without knowing explicitly the task
  - One-shot task transfer: produce output for a task being shown one example of the task as additional input
  - Few-shot task transfer: produce output for a task being shown few examples of the task as additional input
- It's like giving commands to the network

About GPT-2 and GPT-3

- Key idea: text can also encode commands
  - However it is possible to include specific tokens

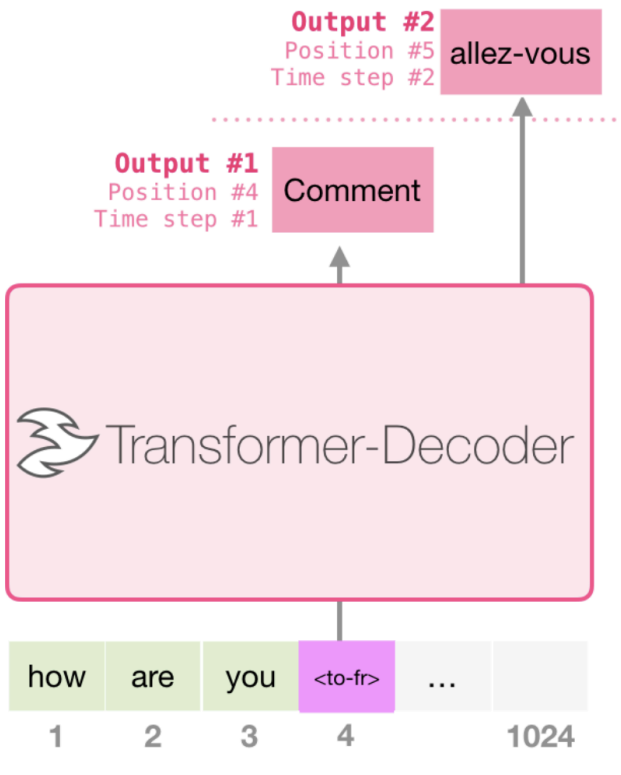


Fine-tuning GPT-2

- Machine translation

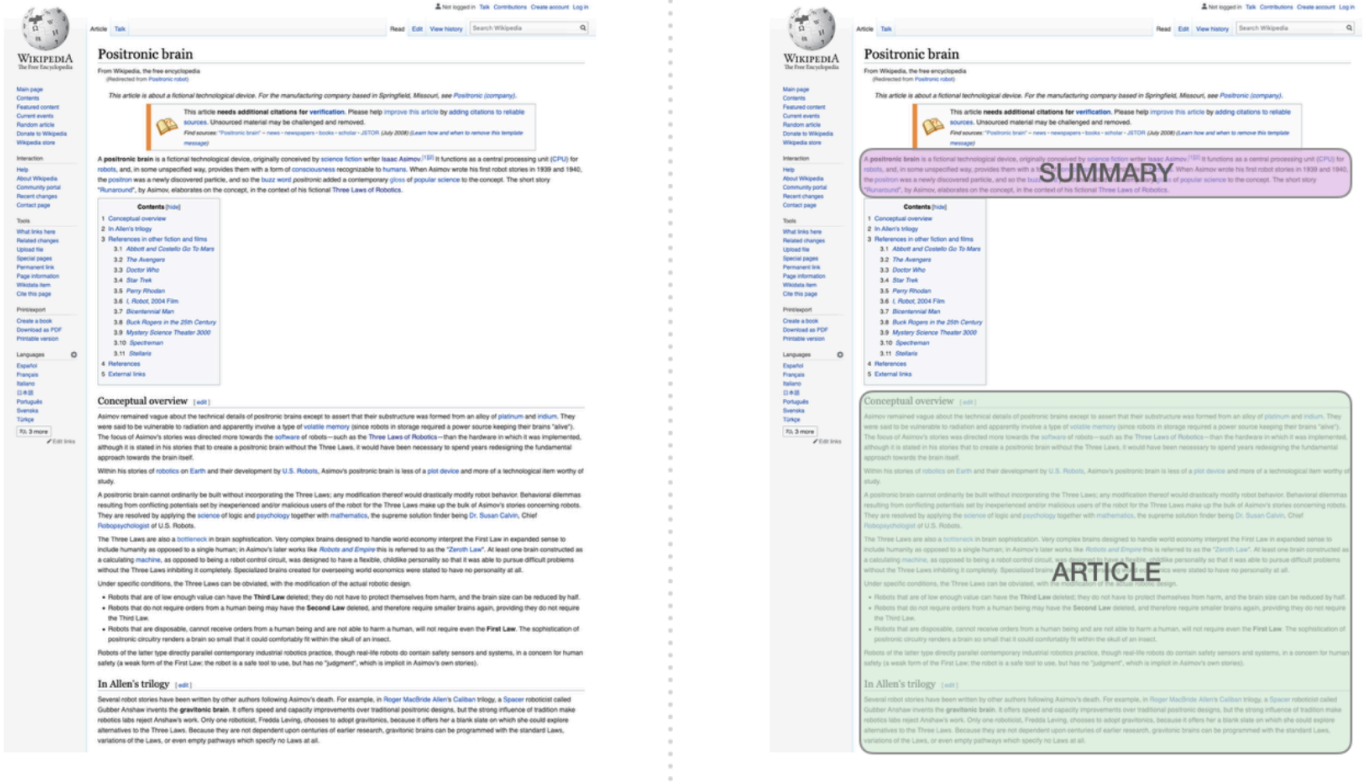
Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



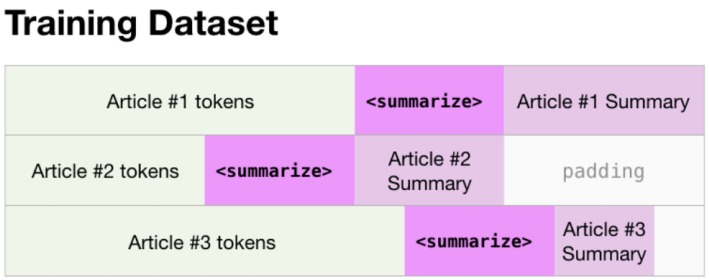
Fine-tuning GPT-2

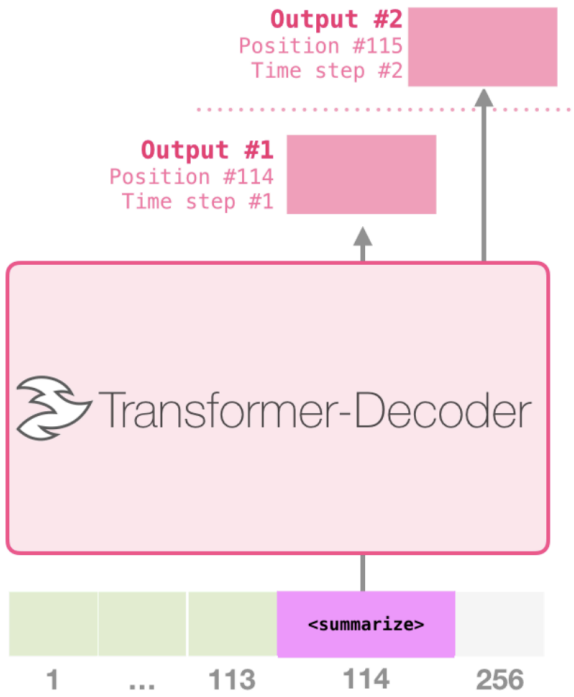
- Summarization



Fine-tuning GPT-2

- Summarization





Audio classification

- Usually the input $\mathbf{X}$ in these cases is given as a sequence of acoustic features (either selected by hand or transferred for deep models)
- Any kind of layer can be used for this input
  - 1D convolutions and pooling
  - (Bidirectional) Recurrent
  - Attention
- The difference is on how the last layers are treated
  - It depends on the problem
- The aforementioned layers produce a continuous output, directly compatible with continuous labelling
  - Just add a fully connected layer sliding over input, or use either a 1D CNN or a (Bi)RNN all with $\mathrm{softmax}()$ activation
- To deal with classification is necessary to flatten information along the time axis
  - It's possible to do so with GAP, MAP, taking the last state of the RNN or having an attention layer with a single query vector (also by fixing input length and unrolling the feature map, but not so elegant and transferable)
  - Finally put a fully connected layer with $\mathrm{softmax}()$ activation on top of it
- Can be treated in the same way of text conceptually

TEXT ANALYSIS

POS tagging and NER

- Part-Of-Speech (POS) tagging and Named Entity Recognition (NER) are sequence labelling problems over text
- Use the layers we discussed before
  - 1D CNN are fast a yield good results but the receptive field is limited
  - BiRNN allow to take all other words in the sentence as context but are slow in computation
  - Attention is fast and use all the context but it requires many computations
  - Fine-tune a pertained transformer

POS tagging and NER

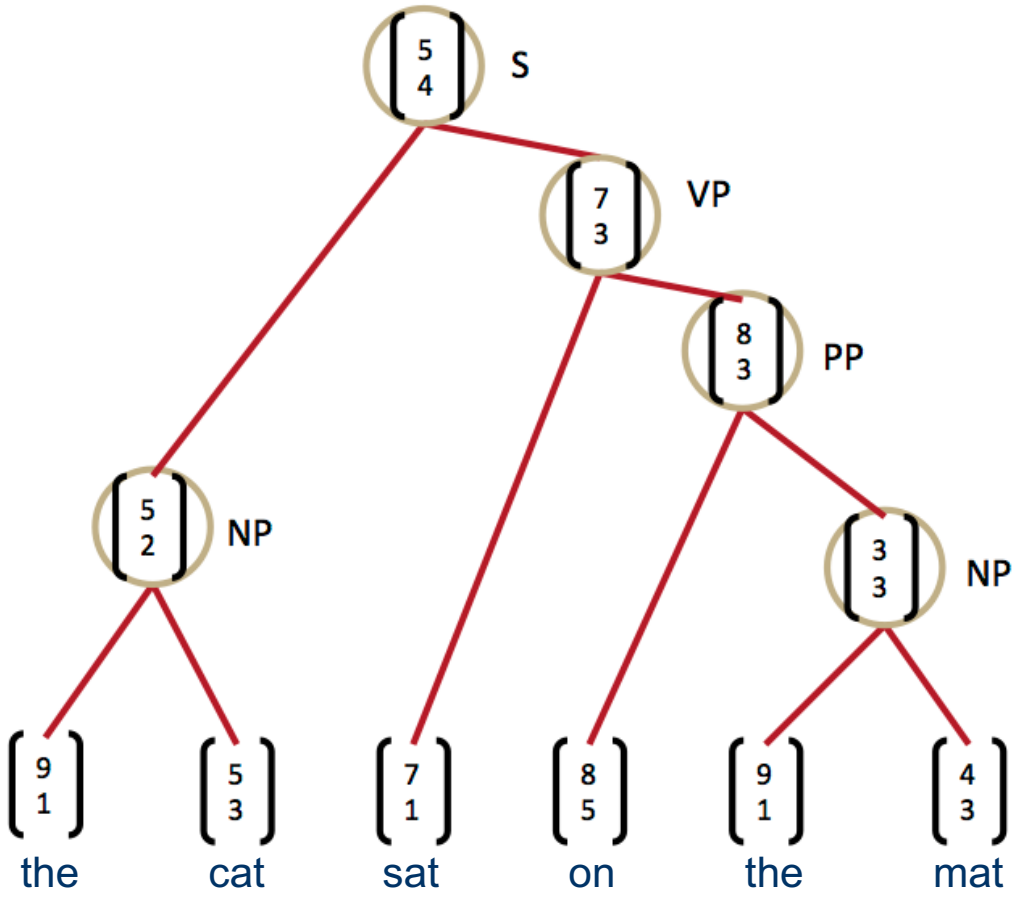
- In general
  - Input $\mathbf{X} \in \mathbb{R}^{(t \times d)}$ where each element is a word embedding
  - Output $\mathbf{Y} \in \mathbb{R}^{(t \times \|\mathscr{C}\|)}$ where $\mathscr{C}$ is the set of POS tags or NER classes
    - Each element of the output $\mathbf{y}_{(k \in [1,t])}$ is computed as a $\mathrm{softmax}()$ over the possible classes
- Train the network to minimize the cross entropy over each element of the output

These slides on parsing
are taken from
https://nlp.stanford.edu/courses/NAACL2013/

Parsing

- Parsing bit more complex than classic problems
  - Data structure is not a sequence but a tree
- Starting from word embeddings
  - Generate the parse tree
  - And the embedding of the whole sentence
- Use Recursive Neural Networks for structure prediction
  - In practice learn a parser

Parsing



Parsing

- Predicting structure
  - Input: two candidate children embeddings
  - Outputs:
    - The embedding representation of the two merged nodes (computed from their concatenation $\left[\mathbf{h}_{(k)}^{\top} \quad \mathbf{h}_{(k+1)}^{\top}\right]^{\top}$)

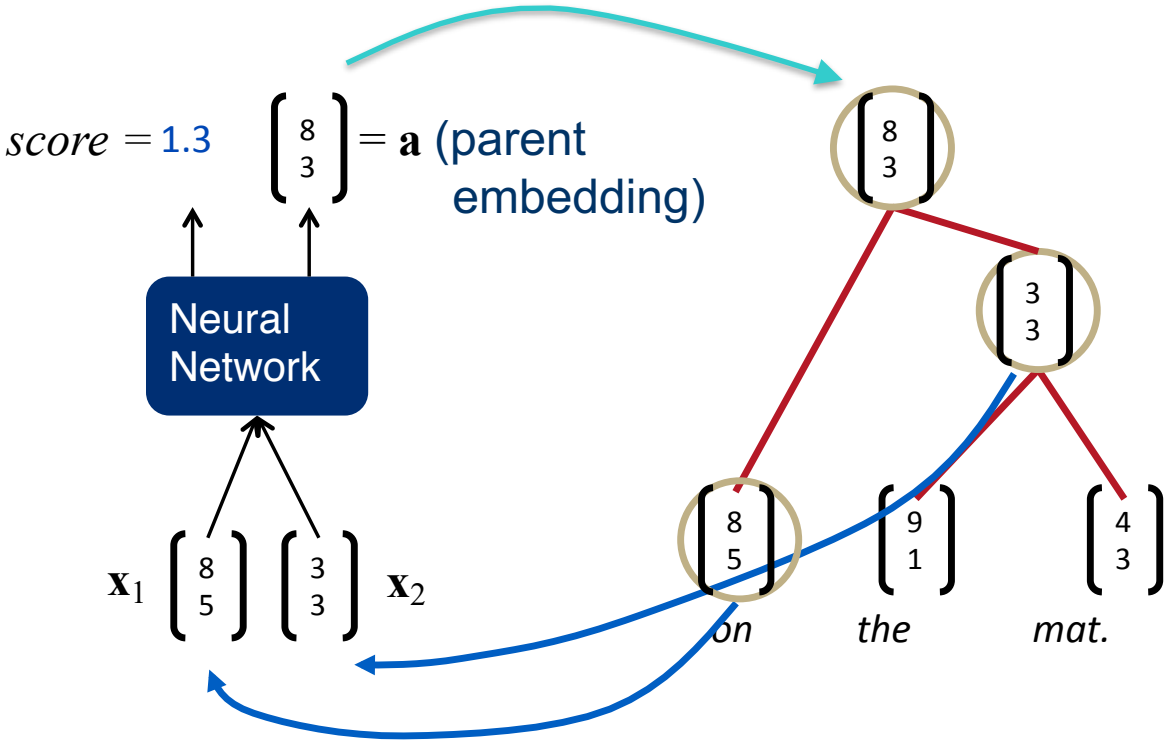
      $$\mathbf{h}'_{(k')} = f_{\text{activation}}\left(\mathbf{W}^{\top} \cdot \begin{bmatrix} \mathbf{h}_{(k)} \\ \mathbf{h}_{(k+1)} \end{bmatrix} + \mathbf{b}\right)$$

    - A score measuring how plausible he new node is (computed from the combination)

      $$\mathbf{s}_{(k,k+1\rightarrow k')} = \mathbf{W}_{\text{score}}^{\top} \cdot \mathbf{h}'_{k'}$$

Parsing

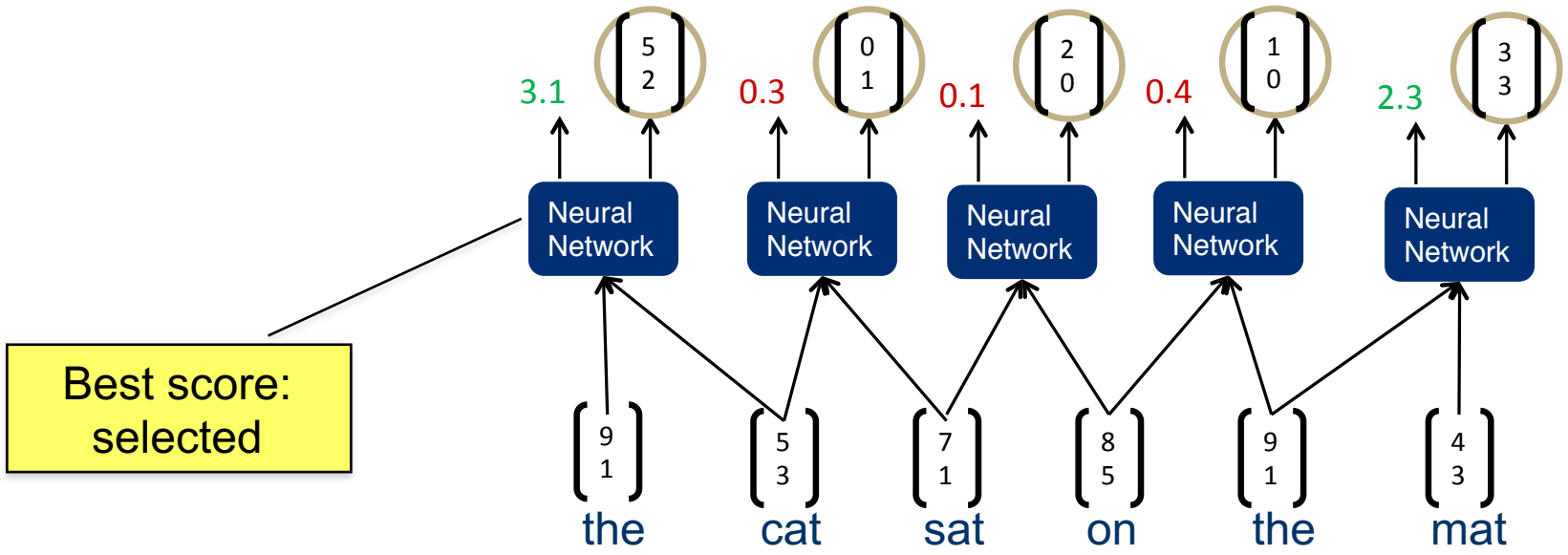
- Predicting the structure node by node



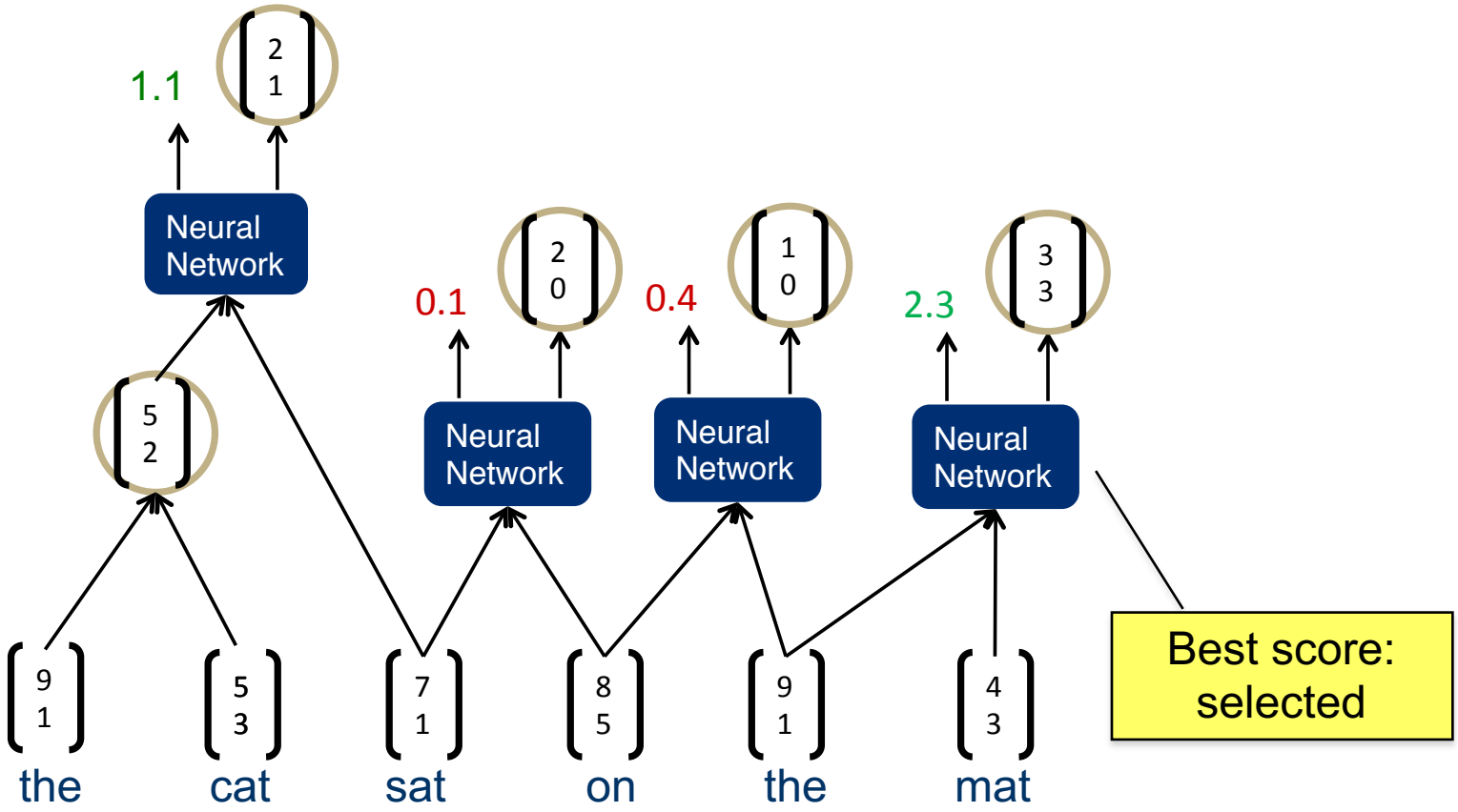
Parsing

- Predicting structure
  - At each step (iterate until all nodes are merged)
    - Compute the merging score of all possible pair of nodes in the input
    - Substitute only the couple with the highest score with its combination

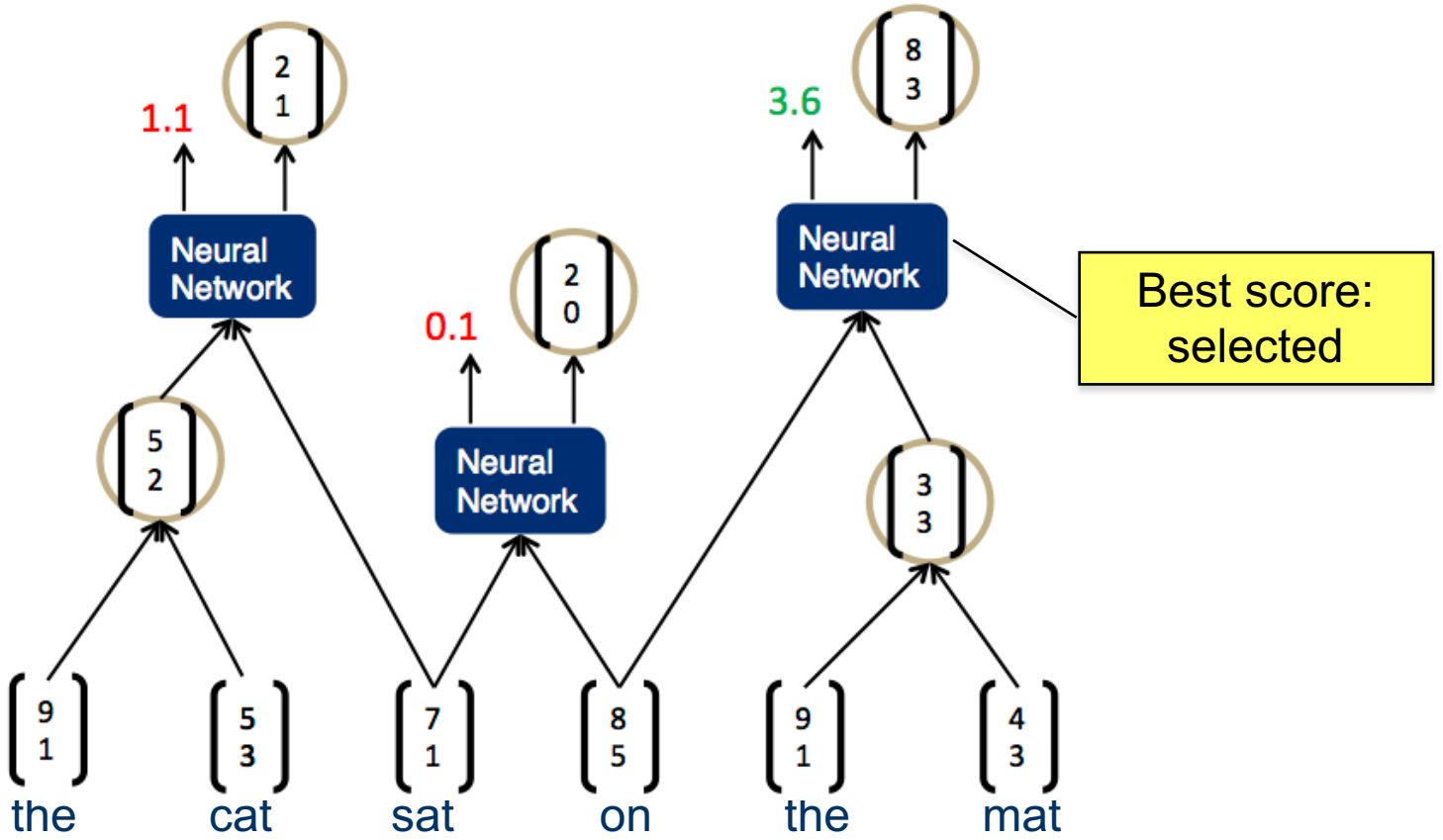
Parsing


Best score: selected
3.1
0.3
0.1
0.4
2.3
Neural Network
Neural Network
Neural Network
Neural Network
Neural Network
the
cat
sat
on
the
mat

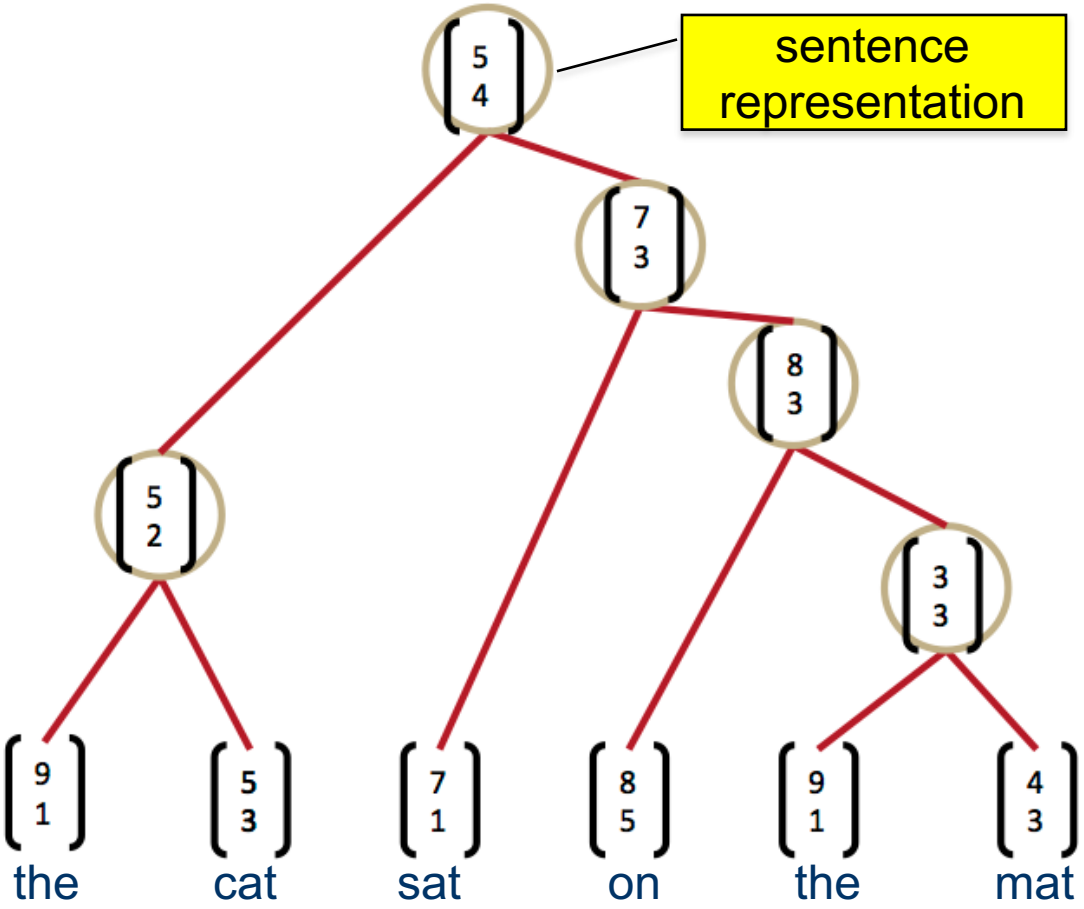
Parsing


1.1
2
1
Neural Network
5
2
0.1
2
0
Neural Network
0.4
1
0
Neural Network
2.3
3
3
Neural Network
9
1
5
3
7
1
8
5
9
1
4
3
the
cat
sat
on
the
mat
Best score: selected

Parsing


1.1
Neural Network
0.1
Neural Network
3.6
Neural Network
Best score: selected
the
cat
sat
on
the
mat

Parsing


5
4
sentence representation
7
3
8
3
5
2
3
3
9
1
5
3
7
1
8
5
9
1
4
3
the
cat
sat
on
the
mat

Parsing (max-marging framework)

- The score of a tree $\mathcal{Y}$ is computed as the sum of the parsing decision scores at each node of $\mathcal{Y}$
- Supervised learning using samples pairs $(\mathbf{X}, \mathcal{Y})$, where $\mathcal{Y}$: set of nodes (the correct tree) parsing sentence $\mathbf{X}$
- Objective function to minimize:

$$E(\vartheta) = \sum_{y \in \mathcal{Y}} f_{\text{score}}(\mathbf{X}, y) - \max_{y' \in A(\mathbf{X})} \left(f_{\text{score}}(\mathbf{X}, y') + \Delta(y, y') \right)$$

- where:
  - y is a node of the correct parse tree
  - $A(\mathbf{X})$ is the set of possible trees parsing $\mathbf{X}$
  - $f_{\text{score}}(\mathbf{X}, y)$ is the score of node y of correct tree
  - $f_{\text{score}}(\mathbf{X}, y')$ is the score of node y' of a tentative tree
  - The loss $\Delta(y, y') < 0$ penalty for non-existent nodes
- Search in $A(\mathbf{X})$: greedy or beam-search with

Parsing

- Single weight matrix recursive neural network could capture some phenomena but not adequate for more complex, higher order composition and parsing long sentences
- The composition function (score) is the same for all syntactic categories, punctuation, etc.
- Idea: Condition the composition function on the syntactic categories, “untie the weights”
  - **Syntactically-Untied** recursive neural network

Parsing

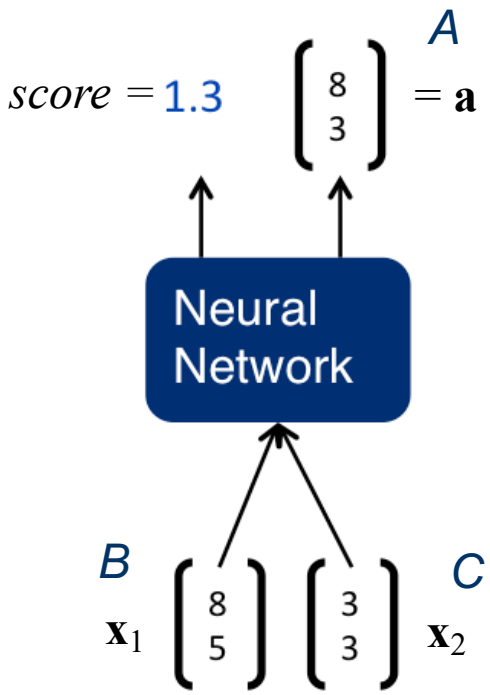
- Allows for different composition functions for pairs of syntactic categories, e.g. Adv + AdjP, VP + NP
- Combines discrete syntactic categories with continuous semantic information
- Problem: Speed. Every candidate score in beam search needs a matrix-vector product
- Solution: Compute score using a linear combination of the log-likelihood from a simple PCFG + RNN
  - Prunes very unlikely candidates for speed
  - Provides coarse syntactic categories of the children for each beam candidate
- **Compositional Vector Grammars**: CVG = PCFG + recursive neural network

Parsing

- Scores at each node computed by combination of PCFG and recursive neural network

$$f_{\text{score}} = \left(\mathbf{u}^{(B,C)}\right)^{\top} \mathbf{a} + \log(p(A \rightarrow BC))$$

- At train time each sample comes with tags B and C on $\mathbf{x}_1$, $\mathbf{x}_2$ and A on parent node
- At run-time, tries tags for $\mathbf{x}_1$, $\mathbf{x}_2$ and parent node (according to some PCFG rule)
- $\mathbf{u}$ depends on B and C
- IInterpretation of the score function:
  - the usual score: continuous model
  - the log-prob of the grammar rule: discrete model

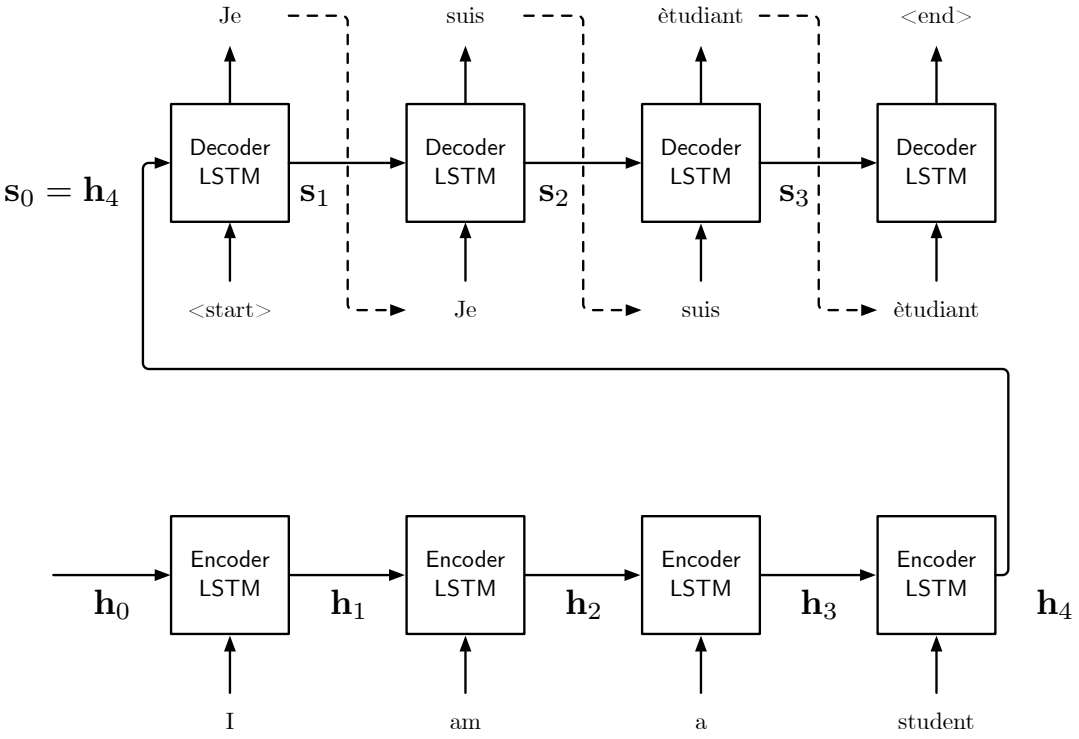


Machine translation

- Given an utterance in one language, automatically translate it to another language
- Usually input $\mathbf{X}$ is the one-hot encoding of the words in the source language, later transformed into word embeddings, the expected output is the probability distribution of the word sequence composing the translation in the target language and the target output $\mathbf{Y}$ is the sequence of one-hot encodings of the words in the translation
- Through the years many solutions have been suggested
  - Recurrent encoder-decoder Seq2Seq
  - Recurrent encoder-decoder Seq2Seq
  - Encoder-decoder Transformer
  - Transformer decoder (remember GPT-2)

Machine translation

- Recurrent encoder-decoder Seq2Seq approach
  - The decoder takes <START> as first input and iterates until <END> is produced as output



Machine translation

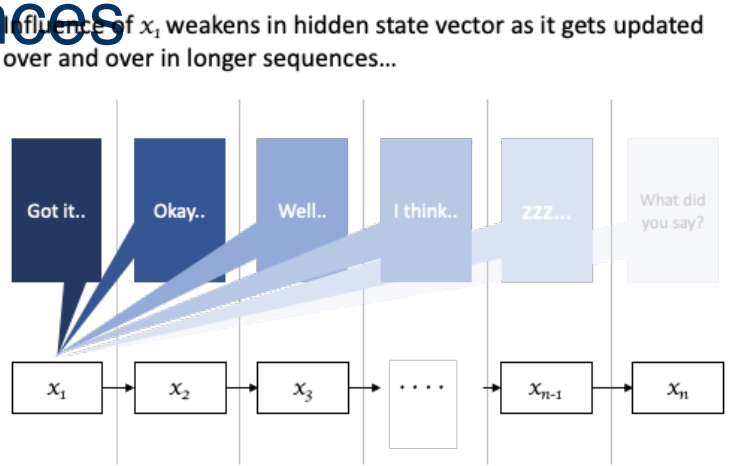
- Predict output as sequence of word probabilities given the input
  - $\mathrm{softmax}()$ over target vocabulary at each step
- Moreover, predict output in autoregressive manner
  - The probability of each word depends on the output words generated up to now
- The network learns the following function

$$\mathbf{Y} = p\left(w_{\mathbf{y}_{(1)}}, \ldots, w_{\mathbf{y}_{(t')}} \,|\, \mathbf{x}_{(1)}, \ldots, \mathbf{x}_t\right) = \prod_{k=1}^{t'} \mathbf{y}_{(k)} = \prod_{k=1}^{t'} p\left(w_{\mathbf{y}_{(k)}} \,|\, \mathbf{X}, \mathbf{y}_{(1)}, \ldots, \mathbf{y}_{(k-1)}\right)$$

  - Output at each step is the probability to observe a certain word of the sequence
- The output is given decoding the probabilities
  - **Greedy decoding**: at each step take only the most probable word
  - **Beam-search decoding**: at each step consider the first n most probable sequences generate up to now (explore other possibilities)

Machine translation

- First solution used LSTMs for encoder and decoder and fed last encoder hidden state as decoder initial state
  - Not capable of dealing with long sequences

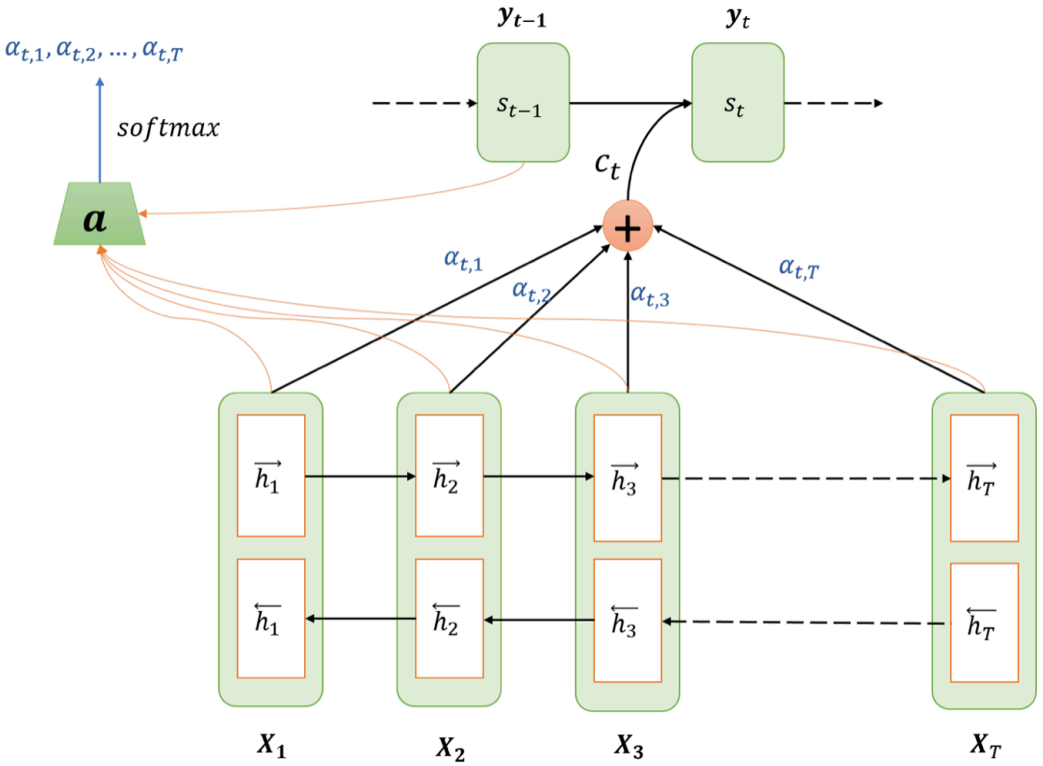


Machine translation

- To help deal with long sequences attention between encoder and decoder was introduced
- Input to decoding is
    - Previous hidden and memory states
    - Concatenation between latest symbol generated and the encoder-decoder attention
        - Encoder hidden states take the role of keys and values, decoder latest hidden state is the query

Machine translation

- Recurrent encoder-decoder Seq2Seq with attention

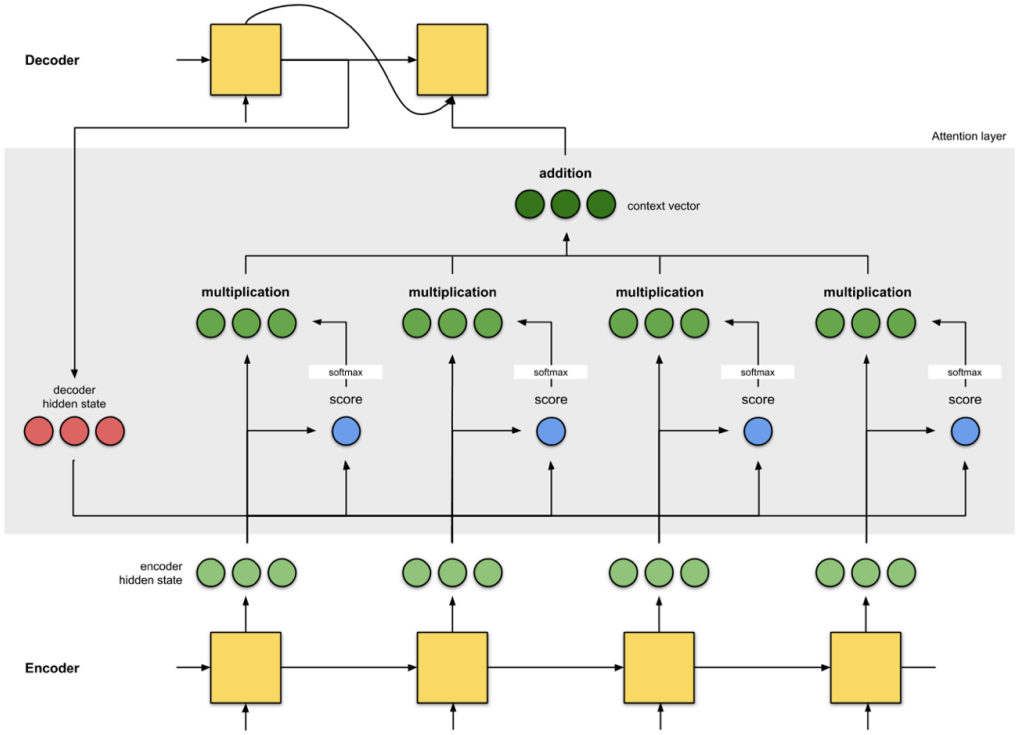


Machine translation

- Visualizing attention mechanism computation

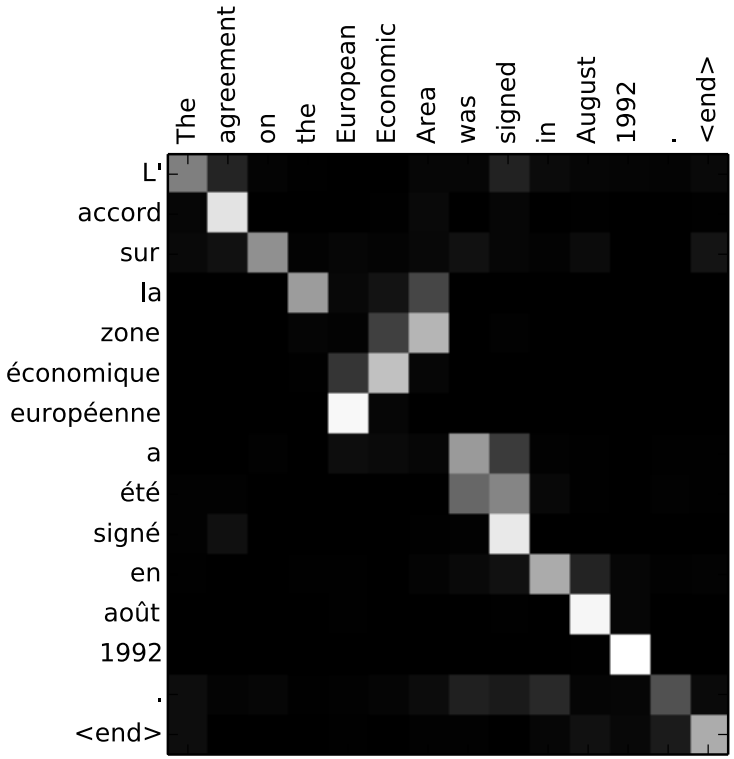
Machine translation

- Visualizing attention mechanism computation



Machine translation

- Visualizing encoder-decoder attention

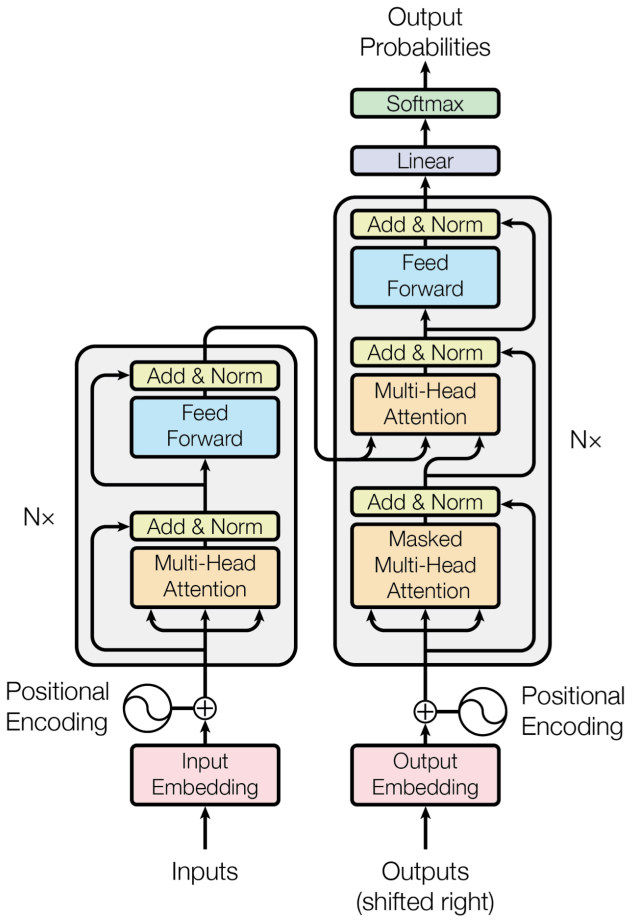


Machine translation

- Actually as long as there is some positional encoding on the input and the output it can be all done through attention layers
  - Self-attention in the encoder
  - Masked self-attention in the decoder
  - Encoder-decoder attention between encoder and decoder

Machine translation

- Transformer architecture for machine translation



Question answering

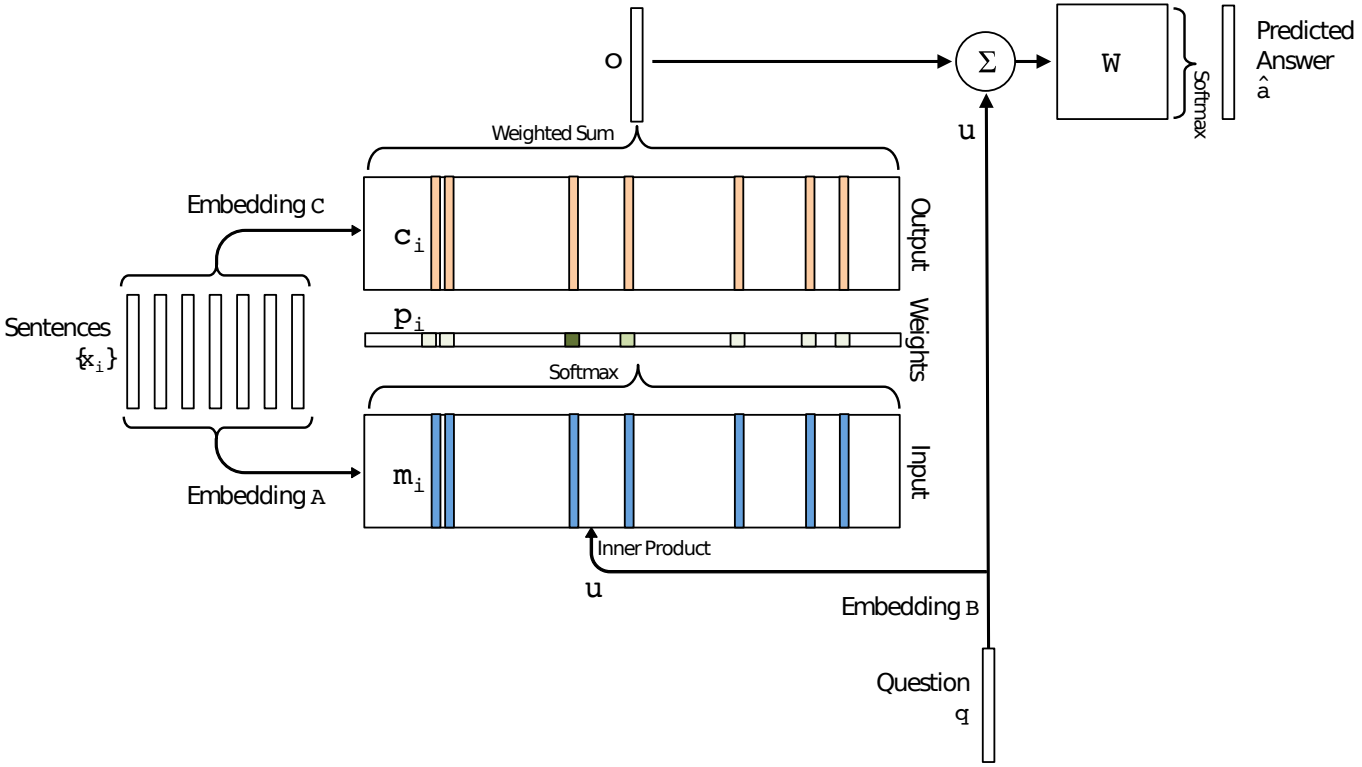
- Given a sequence of sentences and a final question answer the question
- Through the years many solutions have been suggested
  - Some recurrent Seq2Seq
  - Memory networks
    - First time attention mechanism was used
  - Transformer architectures (remember GPT-2 generative capabilities)

Question answering

- End-to-end memory networks
  - Leverage attention to understand with parts of the input sentences $\mathcal{X}$ need to be used to answer question $\mathbf{q}$ and generate answer $\hat{\mathbf{a}}$ that matches the target one $\mathbf{a}$
  - Use 2 additional memory buffers keys buffer $\mathcal{K}$ and values buffer $\mathcal{V}$
- Inputs, question and output are sentences treated as vector
  - Authors used a modified BOW representation
    - They took into account word order and sentences order
    - This is done at embedding time
    - Requires additional parameters a part from those reported
  - All the considered elements are projected into embedding spaces though linear transformations

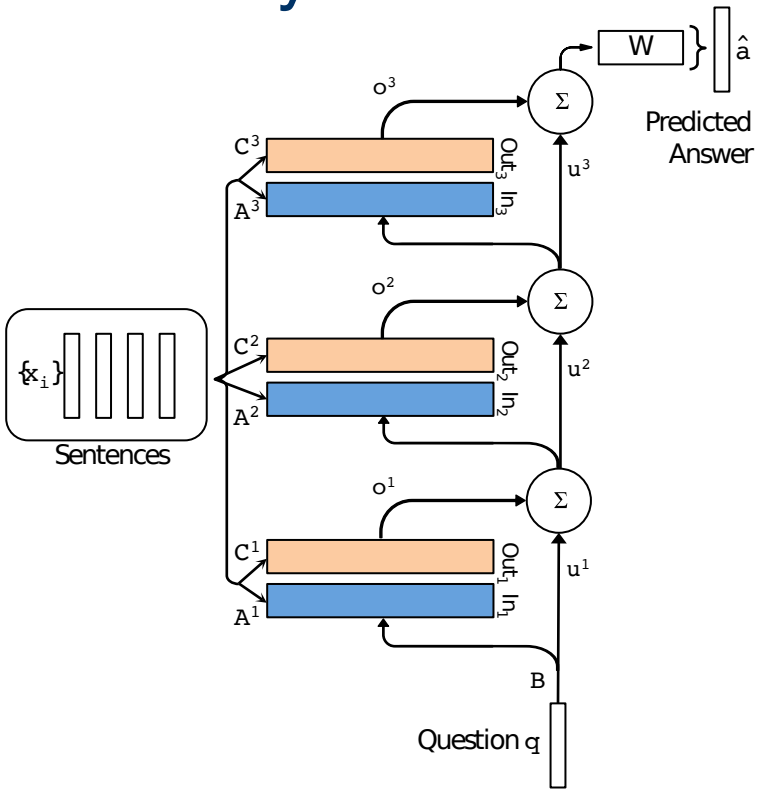
Question answering

- End-to-end memory network architecture



Question answering

- End-to-end memory network architecture



Question answering

- Inputs $\mathbf{x}$ ($\|\mathbf{x}\| = \|\mathscr{V}\|$) are projected into the key and value embedding space
  - $\mathbf{k} = \mathbf{W}_k^\top \cdot \mathbf{x}$ with $\mathbf{k} \in \mathbb{R}^d$ and $\mathbf{W}_k \in \mathbb{R}^{(\|V\| \times d)}$
  - $\mathbf{v} = \mathbf{W}_v^\top \cdot \mathbf{x}$ with $\mathbf{v} \in \mathbb{R}^d$ and $\mathbf{W}_v \in \mathbb{R}^{(\|V\| \times d)}$
- Query $\mathbf{q}$ ($\|\mathbf{q}\| = \|\mathscr{V}\|$) is projected into the query embedding space
  - $\mathbf{u} = \mathbf{W}_u^\top \cdot \mathbf{q}$ with $\mathbf{k} \in \mathbb{R}^d$ and $\mathbf{W}_k \in \mathbb{R}^{(\|V\| \times d)}$

Question answering

- To answer question compute attention scores between query and keys

  $p_i = f_{\text{score}}(\mathbf{x}_i, \mathbf{q}) = \text{softmax}(\mathbf{u}^\top \cdot \mathbf{k}_i)$
- Use the results to produce a weighted combination of values

  $\mathbf{o} = \sum_i p_i \mathbf{v}_i$
- Compute the output through a linear transformation over the sum query embedding and weighted combination followed by $\text{softmax}()$ activation

  $\hat{\mathbf{a}}_i = \text{softmax}\left(\mathbf{W}^\top \cdot (\mathbf{u} + \mathbf{o})\right)$

Chatbots

- In the chatbot taxonomy, neural network based ones fall in the data drive category
- Neural networks can be used for both
  - **Information Retrieval** (**IR**) chatbots
    - Retrieve from a fixed corpus the most probable response given the latest prompt
  - **Generative** chatbots
    - Directly generate the answer to the latest prompt

IR Chatbots

- These are the simplest ones
- Just need an encoding model for the turns and a distance function
  - They key idea is to learn a representation compatible with a similarity function
- Two approaches
  - Retrieve the response that corresponds to the turn in the corpus which is most similar to the current input turn
    $\hat{\mathbf{Y}} = \arg\max\{\mathbf{Y} \in \mathscr{Y}\}\sigma_{\text{matching}}(\mathbf{Y}, \mathbf{X}) = Y' \in \arg\max\{(\mathbf{X}', \mathbf{Y}') \in \mathscr{D}\}\sigma_{\text{similarity}}(\mathbf{X}, \mathbf{X}')$
  - Directly look for the response in the corpus which is the most similar to the current turn
    $\hat{\mathbf{Y}} = \arg\max\{\mathbf{Y} \in \mathscr{Y}\}\sigma_{\text{matching}}(\mathbf{Y}, \mathbf{X}) = Y' \in \arg\max\{\mathbf{Y} \in \mathscr{Y}\}\sigma_{\text{similarity}}(\mathbf{X}, \mathbf{Y})$

IR chatbots

- Encode turn with models like those for sentence embeddings
- Use a distance function like cosine similarity to compute the matching score
- To take a wider context into account use a RNN over the turns embedding and use its hidden representation for matching

Generative chatbots

- Generate the response sequence as a sequence of probabilities on the vocabulary
  - At each time step k, the output $\mathbf{y}_{(k)}$ is the probability distribution over the vocabulary, i.e. the probability of observing word w
    - Same as for machine translation
  - Distribution is computed through a $\mathrm{softmax}()$ activation in the last layer
  - Decode with greedy or beam approaches
- More flexible with respect to IR
  - Ideally can adapt to unseen situations, IR cannot provide an answer outside it corpus
- Strongly compatible with Seq2Seq architecturs
  - Benefits well from supervised pre-training over generic text

Generative chatbots

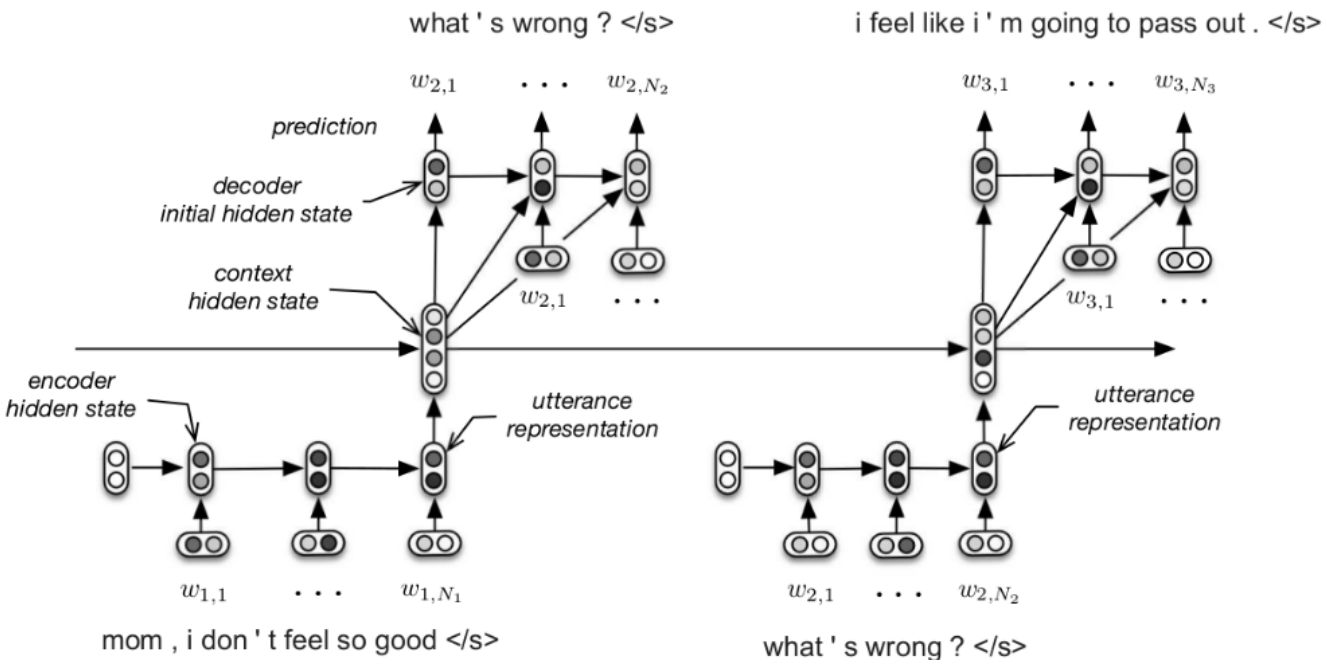
- Approach shifted from RNNs towards Transformers (in particular decoder model)
  - Recurrent vanilla Seq2Seq
  - Recurrent encoder-decoder Seq2Seq
  - Recurrent encoder-decoder Seq2Seq with intermediate context
  - Transformer decoder (Fine tune GPT-like)
    - Compatible with both IR and Generative models

Generative chatbots

- Recurrent encoder-decoder Seq2Seq with intermediate context use context in an interesting manner
  - Encoder produces a single “sentence” representation
  - This is fed to a RNN that predicts the sentence embedding of the answer
  - At each step of decoding the latest generated token is concatenated to the predicted sentence ebmedding
    - In practice it guides decoding

Generative chatbots

- Recurrent encoder-decoder Seq2Seq with context

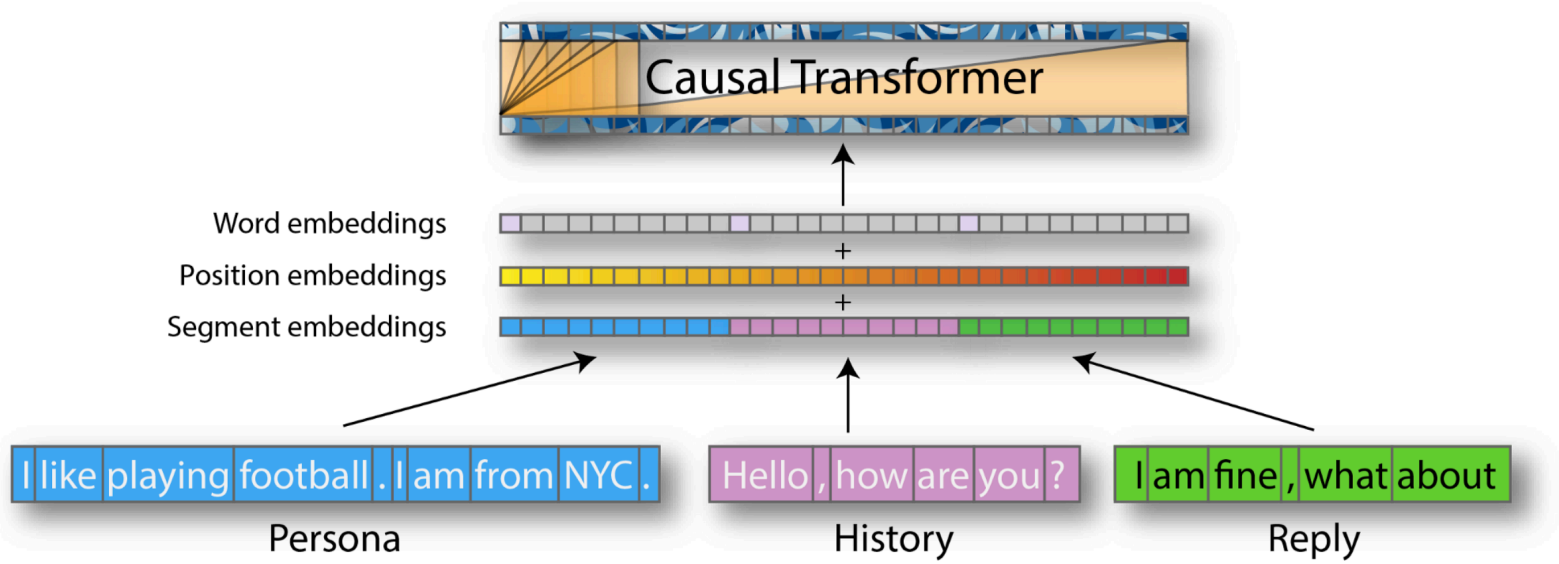


Generative chatbots

- Transformer decoder solution are the most popular nowadays
  - Enable for multi objective training
    - Maximize generative output probability
    - Optimize for positive negative sampling of answers chosen from a pool (including the correct one of course)
  - Can be augmented with many options that can be introduced through natural language
    - E.g. description of participants persona

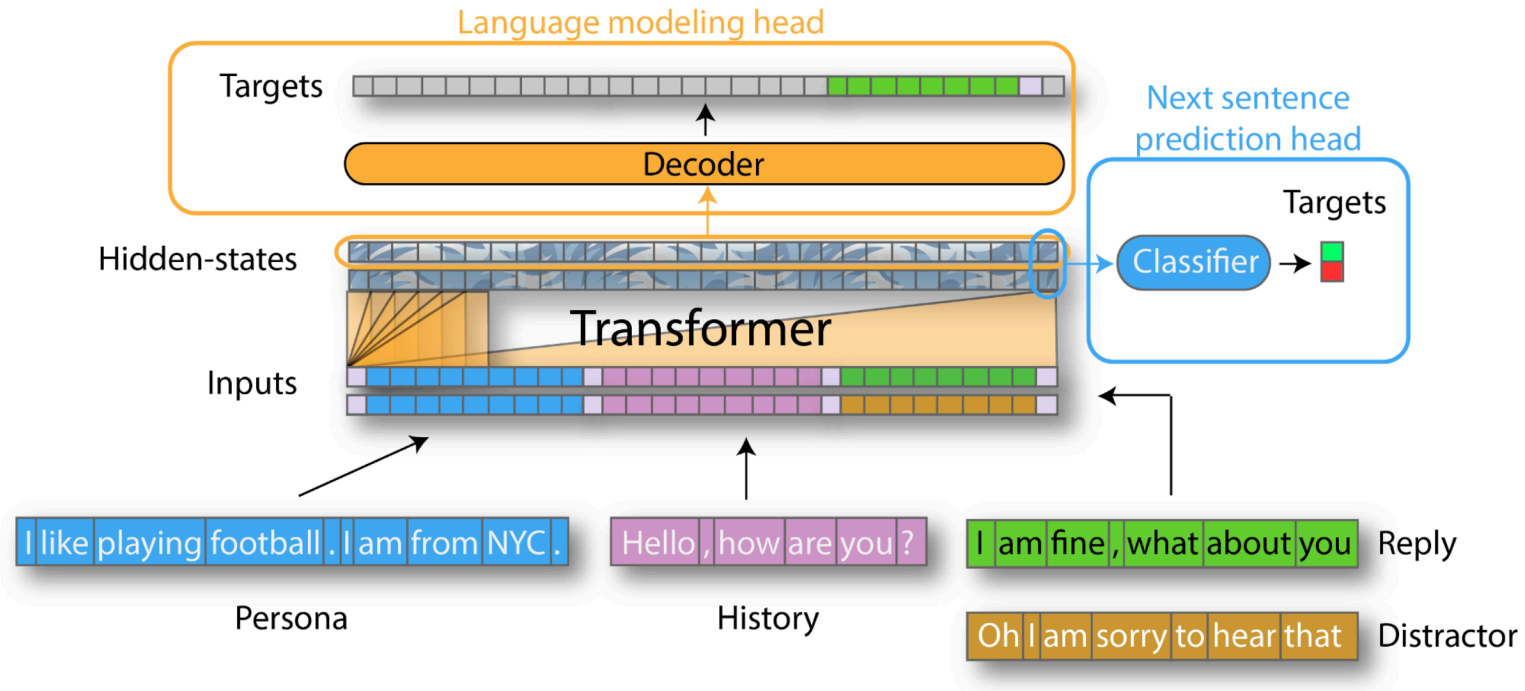
Generative chatbots

- Transformer decoder



Generative chatbots

- Multitask training with transformer



VOICE ANALYSIS

Automatic speech recognition

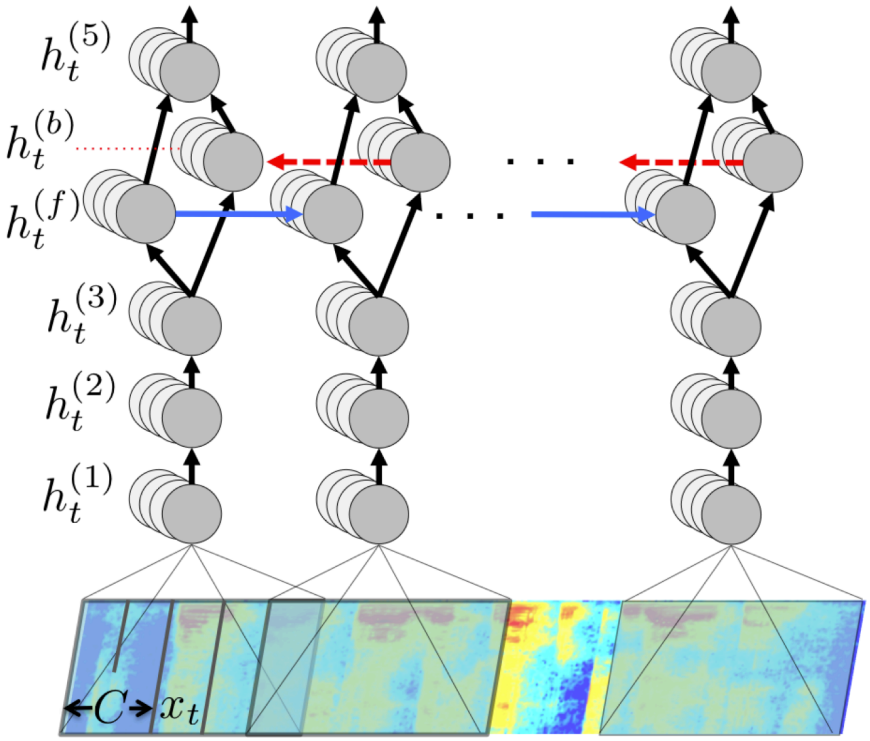
- Automatic Speech Recognition (ASR) improved significantly thanks to neural networks
- Many proposed solutions
- Most used are end-to-end solutions
  - Directly predict the stream of graphemes of phonemes from the input speech signal
- Usually input is given by (Mel) Spectrogram of speech signal

Automatic speech recognition

- Deep Speech is the name of a family of famous neural networks for automatic speech recognition
  - Still state-of-the-art on some speech recognition benchmarks
- Given the input spectrogram $\mathbf{X}$ it directly outputs the probability over the sequence of characters of the transcription $\mathbf{Y}$
  - Each element is computed as a probability distribution over the characters through a $\mathrm{softmax}()$ activation
- In the middle there is a BiRNN (or stack of BiRNNs) and some other layers

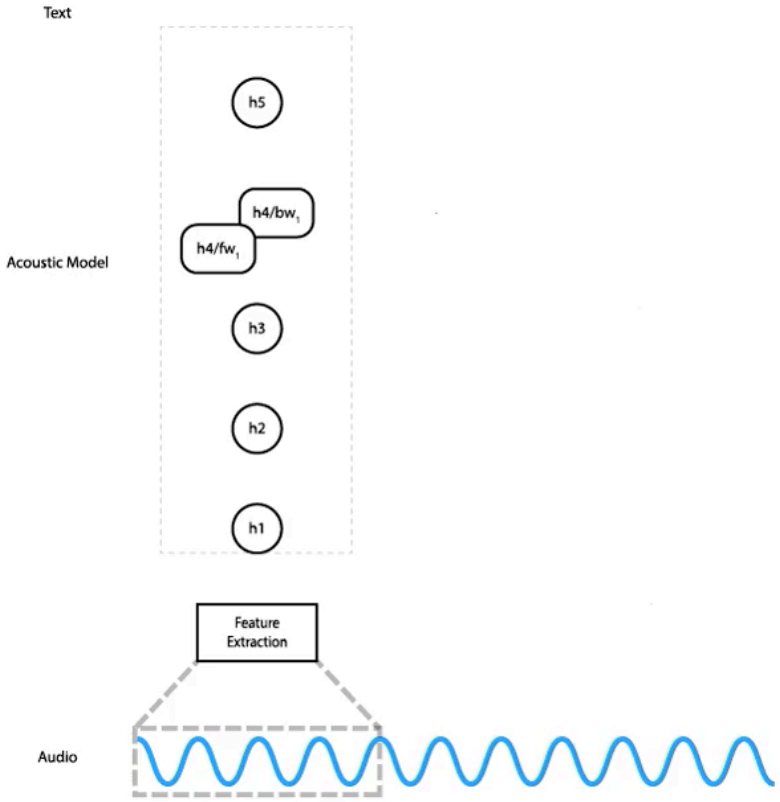
Automatic speech recognition

- Deep Speech architecture



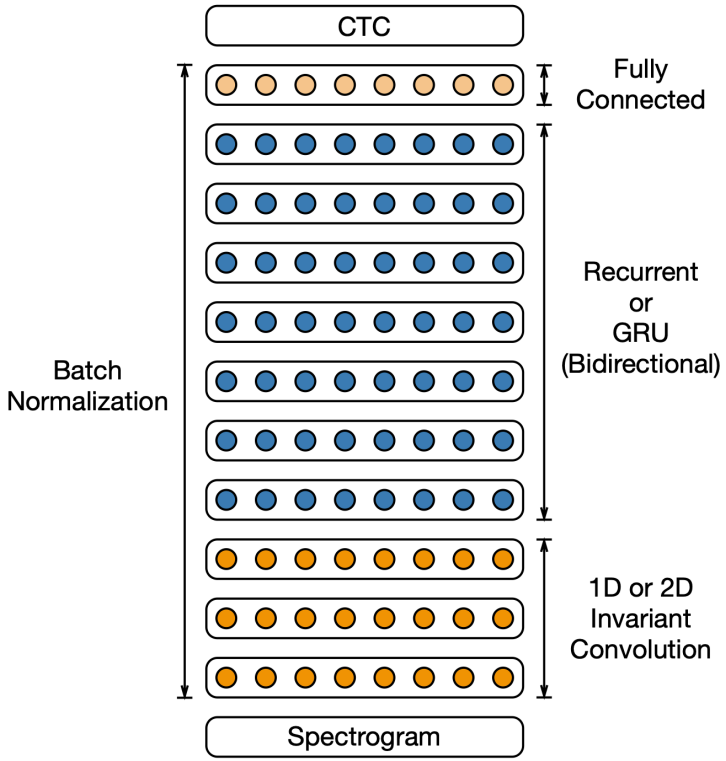
Automatic speech recognition

- Deep speech generating transcription



Automatic speech recognition

- Deep Speech 2 architecture



Automatic speech recognition

- Output at each step is the probability of observing the character in that position of the transcription
  $\mathbf{y}_{(k)} = p(c_{\mathbf{y}_{(k)}} | \mathbf{X})$
- Again decode in a greedy way or with beam search
- Additionally decode with rescoring of the output probabilities though a separate char language model
  - Can be trained separately a much wider corpus
  - Output score is given by
    $Q\left(c_k\right) = \log\left(p(c_{\mathbf{y}_{(k)}} | \mathbf{X})\right) + \alpha \log\left(p_{\text{language model}}(c_{\mathbf{y}_{(k)}} | \mathbf{X})\right) + \beta \text{word count}\,(c)$
  - Ideally retrieve character with highest score

Automatic speech recognition

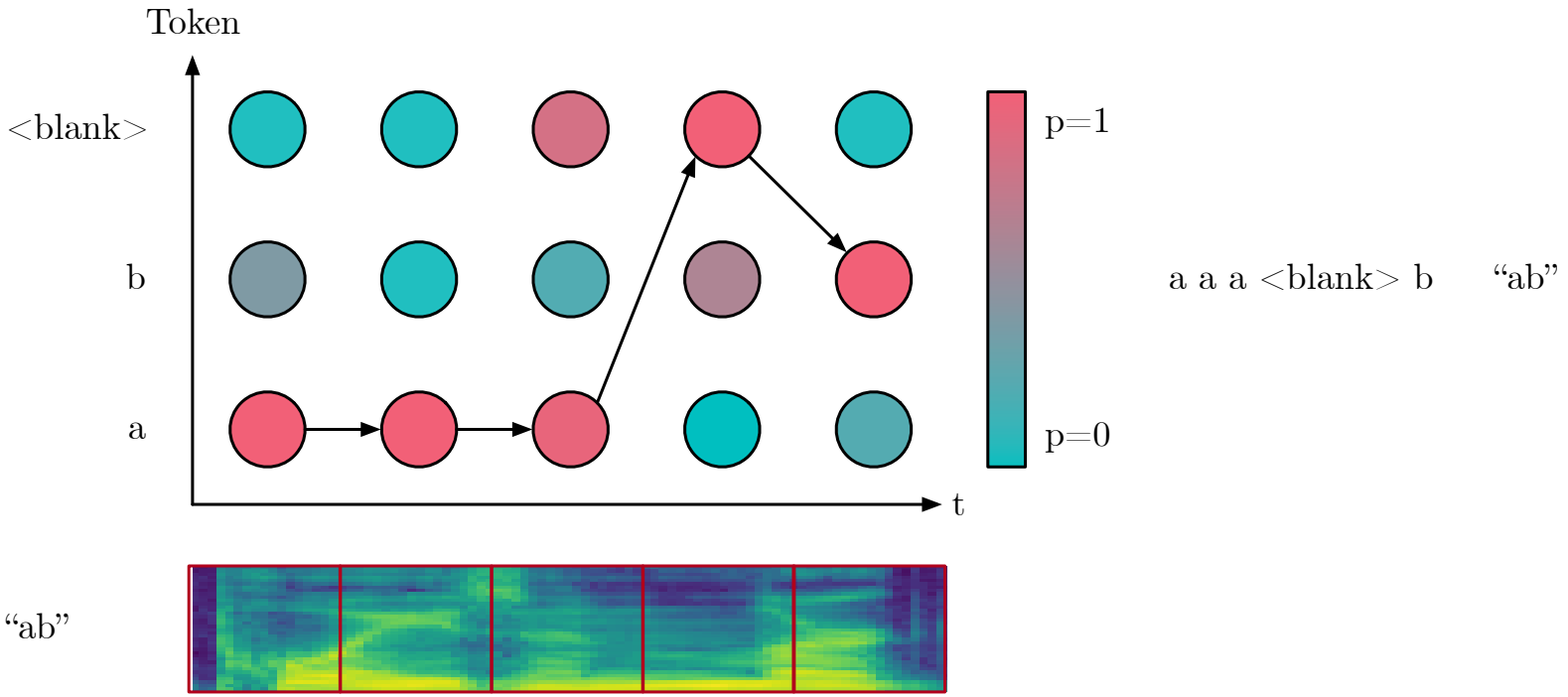
- To avoid the need of carefully segmented and aligned output a particular loss was employed
  - **Connectionist Temporal Classification** (**CTC**)
  - Notice that input spectrogram and output probability stream have different densities, in this way alignment is not required
- The network computes the output considering an additional '<blank>' token
  - The token is removed when decoding the actual characters probability
  - At train time the network learns implicitly to use the blank token to separate the various characters

    a a <blank> b → "ab"

    a a <blank> b <blank> b b a → "abba"

Automatic speech recognition

- CTC decoding



Text-to-speech synthesis

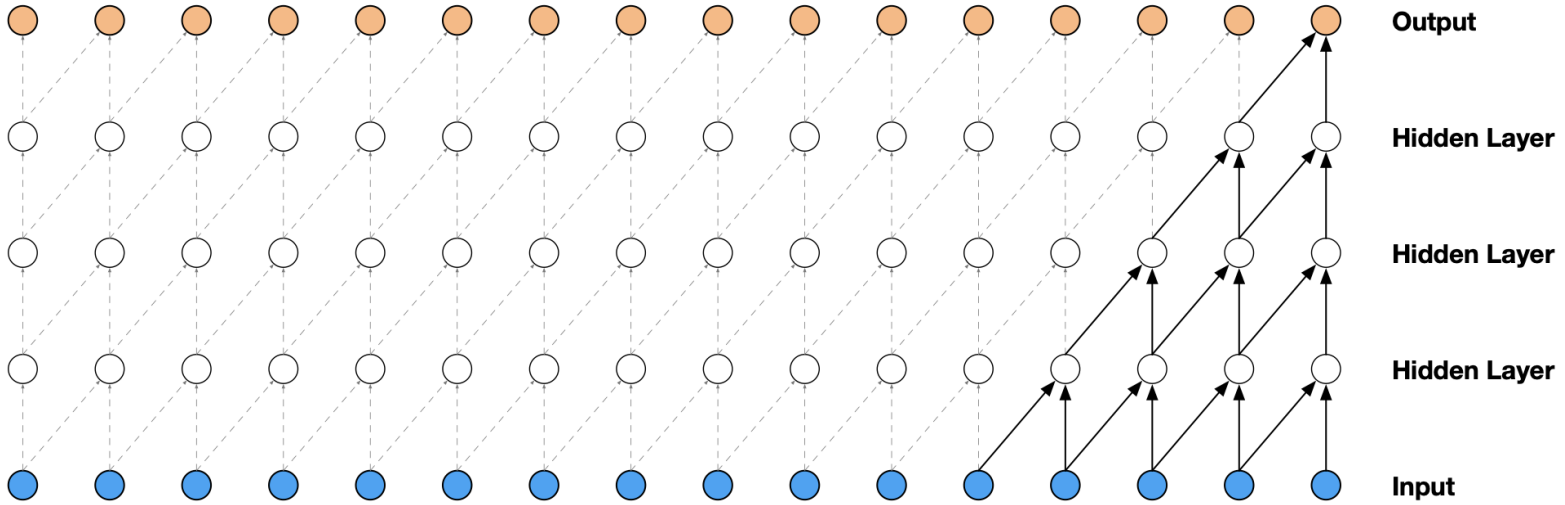
- Given a written text synthesize the speech signal reading it
  - Currently the task is split into two parts
    - Predict (Mel) Spectrogram
    - Use it to condition a vocoder
  - We’re going to see only an end to end architecture to do all at once
- Notice that input and output are swapped with respect to an ASR
  - Can reuse same data sets

Text-to-speech synthesis

- WaveNet is a CNN used to generate raw audio in an autoregressive manner
- Its main input is the one-hot encoding of the quantized input signal
- At each step it uses dilated convolutions to predict the next sample
  - At inference time the predicted sample is consumed to keep on generating the signal
- The network uses causal and dilated convolutions
  - Mixed with a gating mechanism

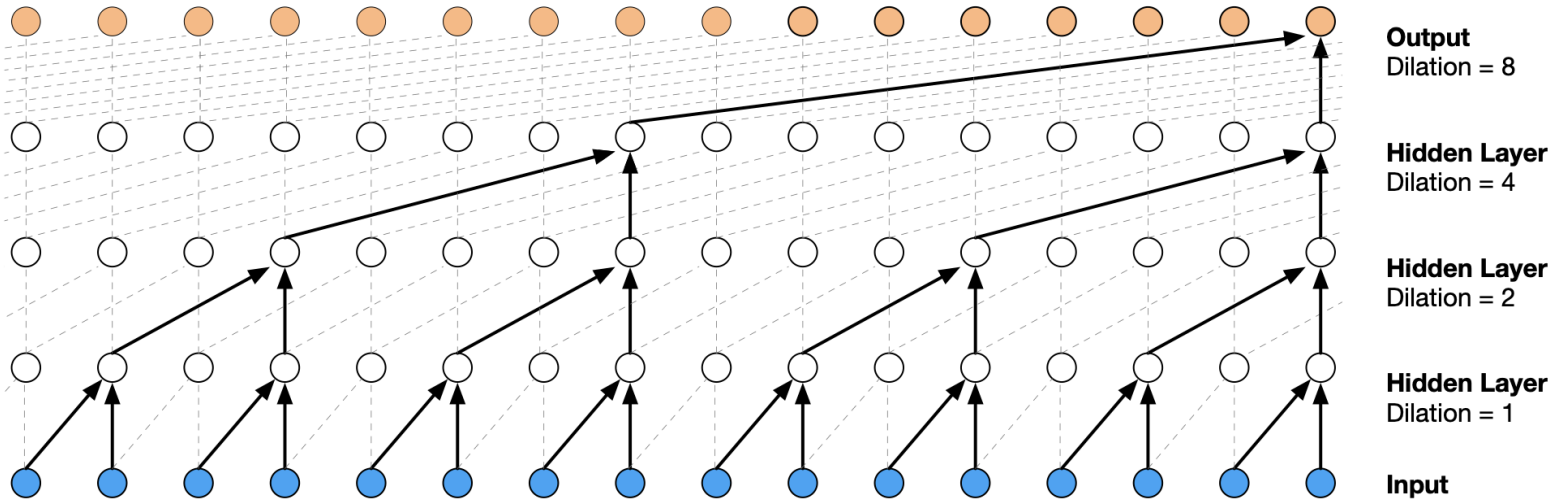
Text-to-speech synthesis

- Causal convolutions



Text-to-speech synthesis

- Dilated causal convolutions



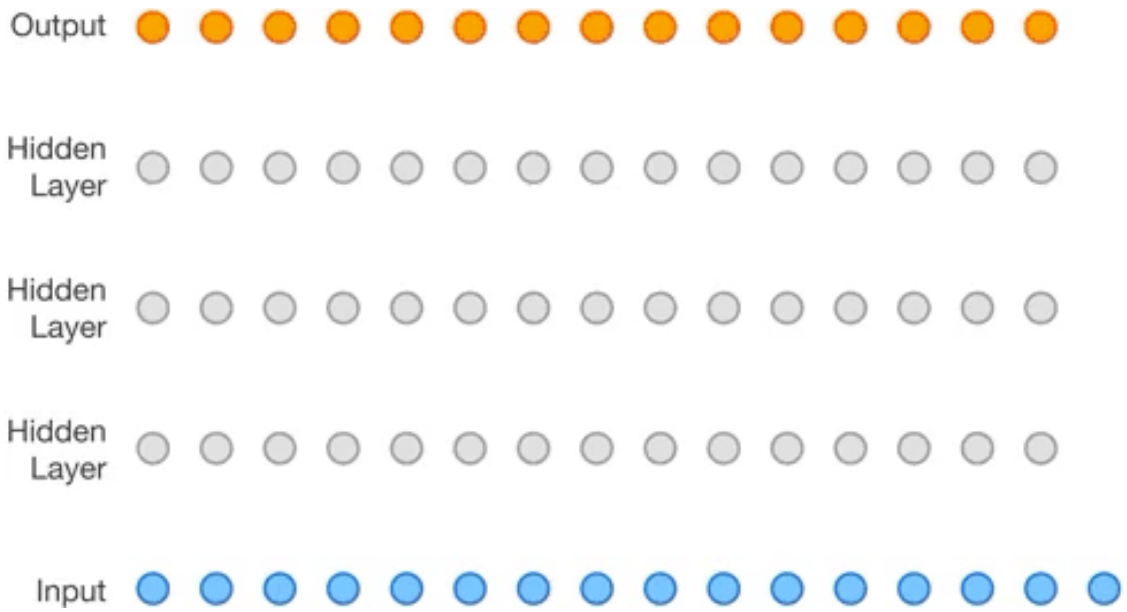
Text-to-speech synthesis

- The output of the network computes the probability of observing next sample given past history and two conditioning (context) feature maps
  - One is for global conditioning $\mathbf{c}_g$ and one for local conditioning $\mathbf{C}_l$
  - The conditions tensors are embeddings representing some global and local informations necessary for the generation process
    - Text to say
    - Speaker style
    - …
- Moreover the output is computed in an autoregressive manner, so the output probability is given by

$$p(\mathbf{X}) = p(\mathbf{X} \mid \mathbf{c}) = \prod_{k=1}^{t} p(\mathbf{x}_{(k)} \mid \mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(k-1)}, \mathbf{c})$$

Text-to-speech synthesis

- Generation process



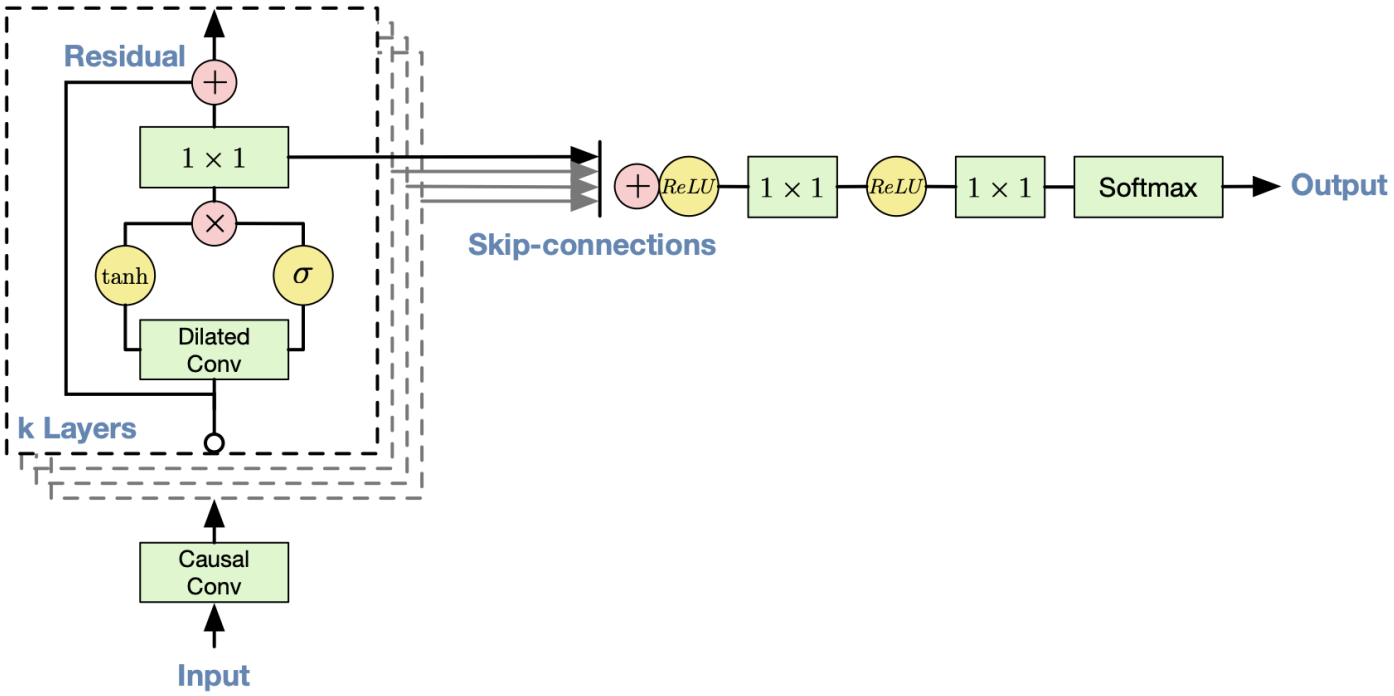
Text-to-speech synthesis

- The hidden feature map at each intermediate layer is given by a gated mechanism combined with the dilated causal convolutions
  - At each layer dilation rate is doubled in order to expand the receptive field
- Gating is controlled through two separate activations

$$\mathbf{Z} = \tanh\left(\mathbf{W}_{f,h} \star \mathbf{H} + \mathbf{V}_{f,c}^{\top} \cdot \mathbf{c}_g + \mathbf{W}_{f,c} \star \mathbf{C}_l\right) \cdot \sigma\left(\mathbf{W}_{g,h} \star \mathbf{H} + \mathbf{V}_{g,c}^{\top} \cdot \mathbf{c}_g + \mathbf{W}_{g,c} \star \mathbf{C}_l\right)$$

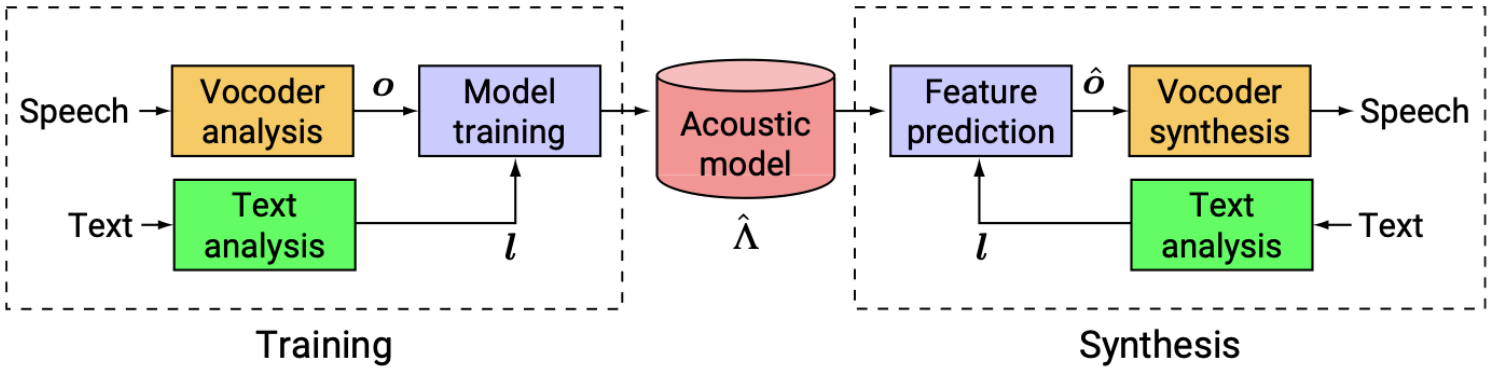
Text-to-speech synthesis

- WaveNet blocks



Text-to-speech synthesis

- Train/inference approach



Speaker diarization

- Given a stream of audio features
- Associate each of the time stamps to its speaker
- No speaker identification
  - Simply separate among those that are speaking in the given audio stream
- Usually treated as an unsupervised/clustering problem
- A model can be trained in a supervised manner using a variant of RNNs

Speaker diarization

- Unbounded Interleaved State RNNs (UIS RNNs) are a variant of RNN designed for diarization
- Given a sequence of audio input features $\mathbf{X} \in \mathbb{R}^{(t \times d)}$ and a sequence of labels identifying the speaker $\mathbf{y} \in \mathbb{N}^t$(defined as a sequence of integers for simplicity)
- UIS RNN are used to model the probability

$$p\left(\mathbf{X}, \mathbf{y}\right) = p\left(\mathbf{x}_{(1)}, y_{(1)}\right) \prod_{k=2}^{t} p\left(\mathbf{x}_{(k)}, y_{(k)} \,|\, \mathbf{x}_{(k-1)}, y_{(k-1)}\right)$$

using an augmented representation

$$p\left(\mathbf{X}, \mathbf{y}, \mathbf{z}\right) = p\left(\mathbf{x}_i^{(1)}, y_i^{(1)}\right) \prod_{k=2}^{t} p\left(\mathbf{x}_{(k)}, y_{(k)}, z_{(k)} \,|\, \mathbf{x}_{(k-1)}, y_{(k-1)}, z_{(k-1)}\right)$$

Speaker diarization

- $z_{(k)} \in 1$ is a binary variable signaling a change of speaker with respect to input frame $k-1$
- Excluding the case $k=1$ the probability can be factorized in

$$p\left(\mathbf{x}_{(k)}, y_{(k)}, z_{(k)} \mid \mathbf{x}_{(k-1)}, y_{(k-1)}, z_{(k-1)}\right) =$$

$$\underbrace{p\left(\mathbf{x}_{(k)} \mid y_{(k)}, \mathbf{x}_{(k-1)}\right)}_{\text{sequence generation}} \underbrace{p\left(y_{(k)} \mid z_{(k)}, y_{(k-1)}\right)}_{\text{speaker assignment}} \underbrace{p\left(z_{(k)} \mid z_{(k-1)}\right)}_{\text{speaker change}}$$

Speaker diarization

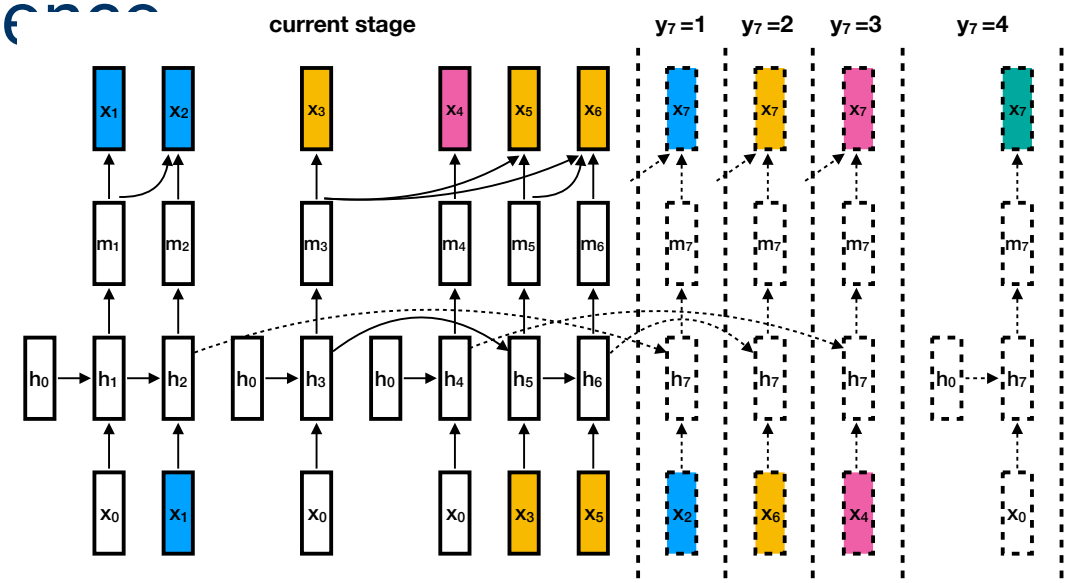
- The key feature of UIS RNN is that when $z_{(k)} = 1$ (i.e. when there's speaker change), they consider for speaker assignment:
  - All the speakers appeared up to that point (excluding the previous one)
  - The possibility of a new speaker
- To do so they store the current hidden state every time there's a speaker change, try to restart from one of the previous and see if there's a speaker change again. If it happens with all the previous the restart from a new hidden state

Speaker diarization

- Assign the first window to speaker 1
- For each following window
  - Predict the probability that the speaker is the same of the previous hidden state
  - If so go on to the next window
  - If that's not the case
    - Save the previous hidden state for the current speaker
    - For each of the remaining speakers saved hidden states predict the probability that the current window does not present a speaker change resorting its last hidden state
    - If all of them fail generate a new speaker

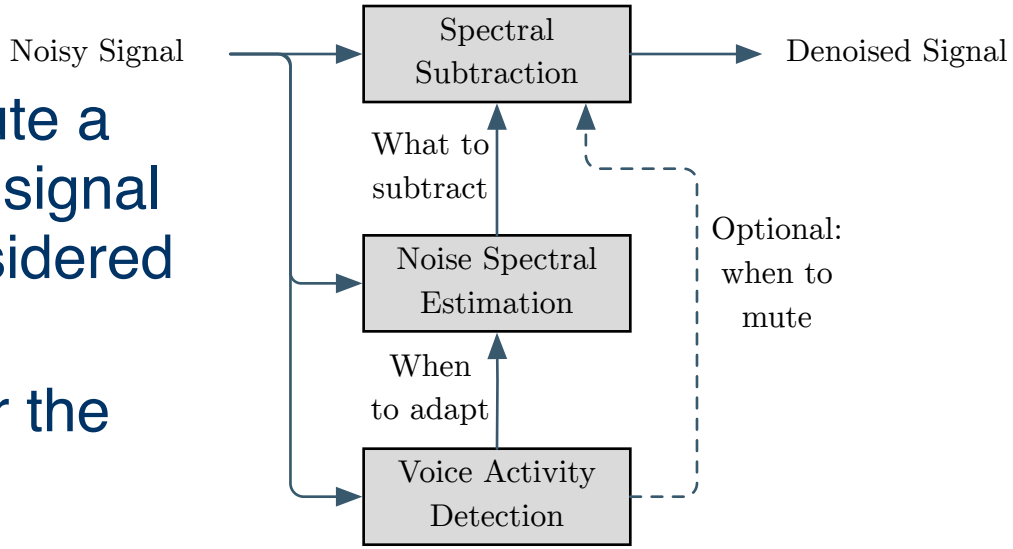
Speaker diarization

- In practice the network just need to learn to predict the value of z given the current input and the currently considered speaker sequence



Noise cancelling

- Regression problem
  - Use a RNN to isolate the spoken parts of a noisy signal
  - At each step compute a gain to apply to the signal for each of the considered bands
  - Sliding window over the input features

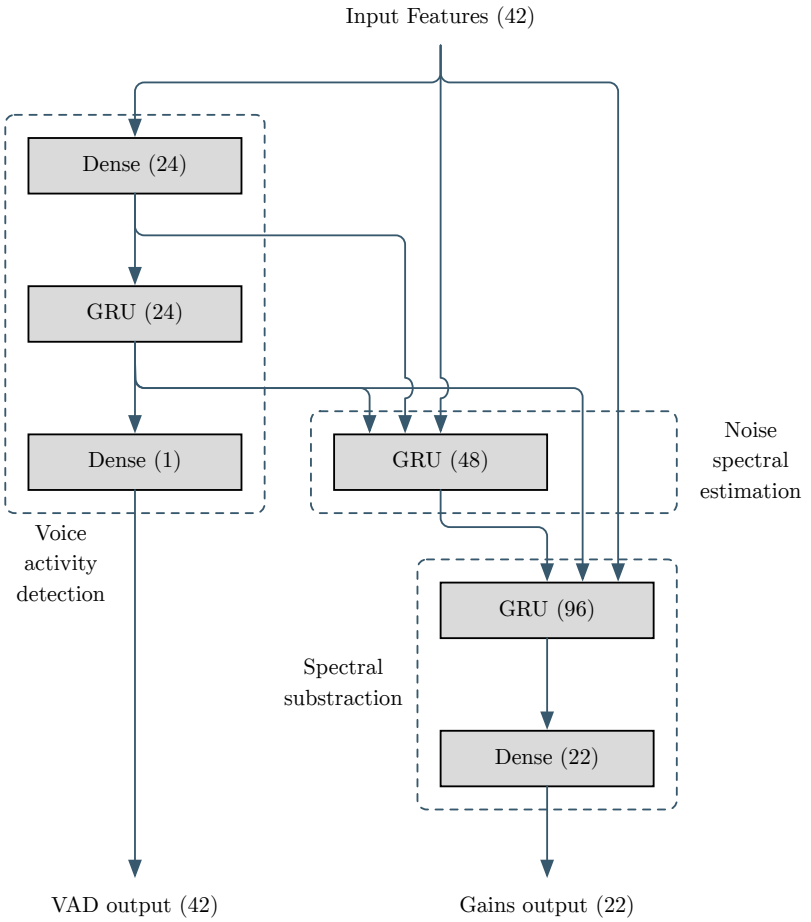


Noise cancelling

- RNNoise learn noise cancelling from a stack of custom features
  - 22 MFCC
  - The first and second derivatives of the first 6 coefficients across frames
  - The pitch period $\frac{1}{F_0}$
  - The pitch gain (voicing strength) in 6 bands
  - A special non-stationarity value that's useful for detecting speech

Noise cancelling

- RNNoise architecture



MULTIMODAL ANALYSIS

Emotion recognition

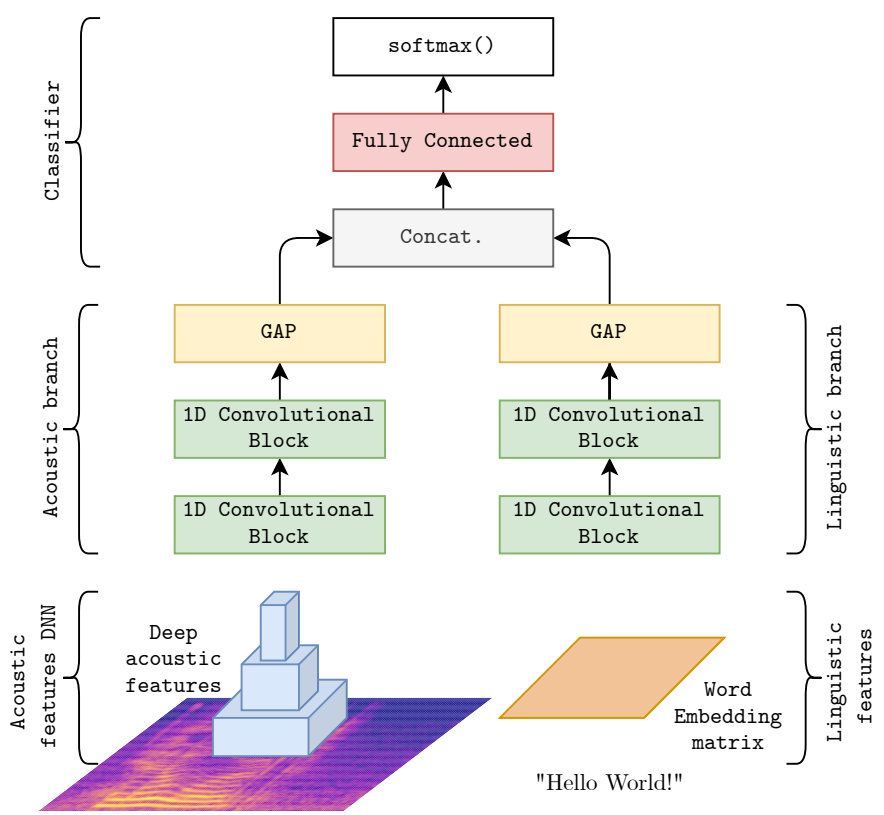
- Emotion recognition is a task that widely benefits from multimodal analysis
  - Use both voice and text
- Simplest solution for these cases is to perform feature fusion
  - Compute transformations over voice and text separately
  - Flatten both along time axis and concatenate the vectors
  - Perform classification through a linear layer with $\mathrm{softmax}()$ activation

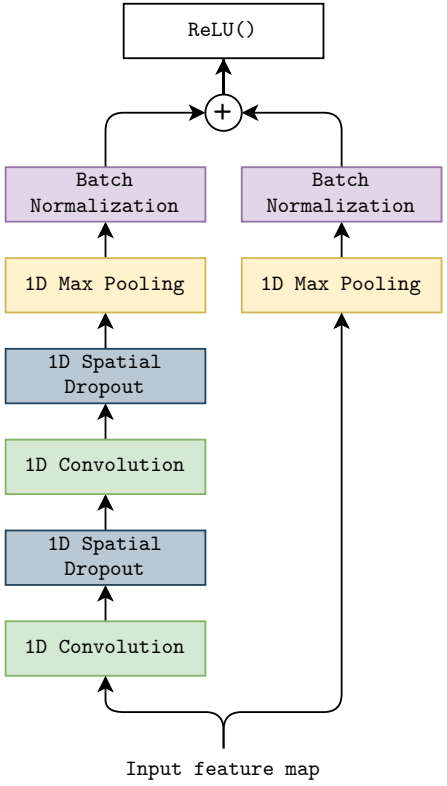
Emotion recognition

- PATHOSnet for emotion recognition
  - Multimodal CNN built on top of pre-trained features
    - VGGish for audio
    - FastText word embeddings for text
  - Flatten both modalities with GAP

Emotion recognition

- PATHOSnet architecture





REFERENCES

References

- https://arxiv.org/pdf/1810.04805.pdf
- https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf
- http://jalammar.github.io/illustrated-gpt2/
- https://www-nlp.stanford.edu/pubs/SocherLinNgManning_ICML2011.pdf
- https://towardsdatascience.com/an-introduction-to-attention-transformers-and-bert-part-1-da0e838c7cda
- https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3
- https://medium.com/@shashank7.iitd/understanding-attention-mechanism-35ff53fc328e
- http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
- https://arxiv.org/pdf/1503.08895.pdf
- https://arxiv.org/pdf/1901.08149.pdf

References

- https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313
- https://arxiv.org/pdf/1507.04808.pdf
- https://arxiv.org/pdf/1412.5567.pdf
- https://arxiv.org/pdf/1512.02595.pdf
- https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate
- https://arxiv.org/pdf/1609.03499.pdf
- https://arxiv.org/pdf/1810.04719.pdf
- https://jmvalin.ca/papers/rnnoise_mmsp2018.pdf