Deep learning for natural language processing
Advanced architectures

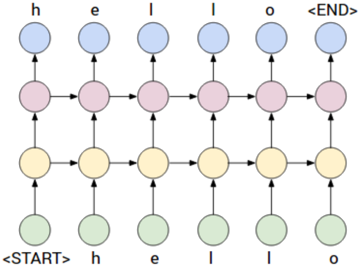
Benoit Favre <benoit.favre@univ-amu.fr>

Aix-Marseille Université, LIF/CNRS

03-2018

Stacked RNNs

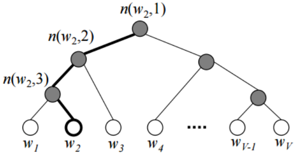
- Increasing hidden state size is very expensive
  - U is of size $(hidden \times hidden)$
  - Can feed the output of a RNN to another RNN cell
  - $\rightarrow$ Multi-resolution analysis, better generalization



- Highway connections create shortcuts between layers
  - $gate_l = \sigma(W_g h_{l-1})$
  - $h_l = \text{LSTM}(h_{l-1}) \odot gate_l + h_{l-1} \odot (1 - gate_l)$

Softmax approximations

- When vocabulary is large (> 10000), the softmax layer gets too expensive
  - Store a $h \times |V|$ matrix in GPU memory
  - Training time gets very long
- Turn the problem to a sequence of decisions
  - Hierarchical softmax

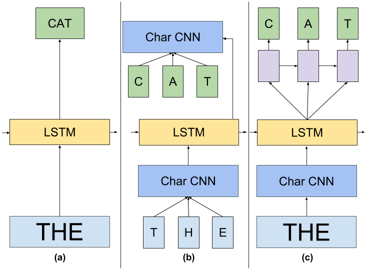


- Turn the problem to a small set of binary decisions
  - Noise contrastive estimation, sampled softmax...
  - $\rightarrow$ Pair target against a small set of randomly selected words
- More here: `http://sebastianruder.com/word-embeddings-softmax/`

Limits of language modeling

- Train a language model on the One Billion Word benchmark
  - "Exploring the Limits of Language Modeling", Jozefowicz et al. 2016
  - 800k different words
  - Best model $\rightarrow$ 3 weeks on 32 GPU
  - PPL: perplexity evaluation metric (lower is better)

System	PPL
RNN-2048	68.3
Interpolated KN 5-GRAM	67.6
LSTM-512	32.2
2-layer LSTM-2048	30.6
Last row + CNN inputs	30.0
Last row + CNN softmax	39.8

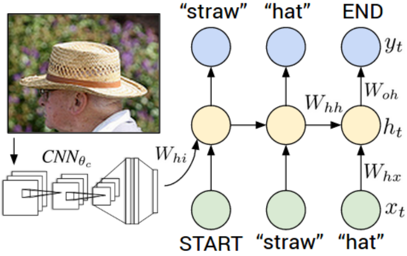


Caption generation

- Language model conditioned on an image
  - Generate image representation with CNN trained to recognize visual concepts
  - Stack image representation with language model input

people skying on a snowy mountain

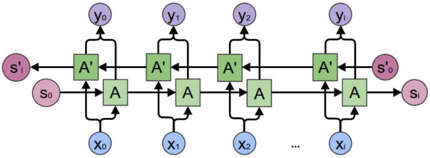
a woman playing tennis



- More here: `https://github.com/karpathy/neuraltalk2`

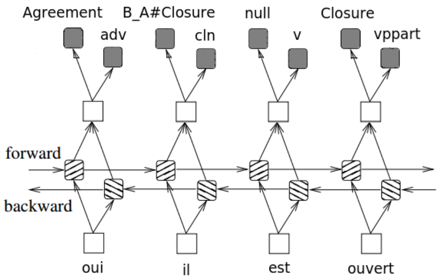
Bidirectional networks

- RNN make predictions independent of the future observations
  - Need to look into the future
- Idea: concatenate the output of a forward and backward RNN
  - The decision can benefit from both past and future observations
  - Only applicable if we can wait for the future to happen



Multi-task learning

- Can we build better representations by training the NN to predict different things?
  - Share the weights of lower system, diverge after representation layer
  - Also applies to feed forward neural networks
- Example: semantic tagging from words
  - Train system to predict low-level and high-level syntactic labels, as well as semantic labels
  - Need training data for each task
  - At test time only keep output of interest



Machine translation (the legacy approach)

Definitions

- $source$: text in the source language (ex: Chinese)
- $target$: text in the target language (ex: English)

Phrase-based statistical translation

- Decouple word translation and word ordering

$$P(target|source) = \frac{P(source|target) \times P(target)}{P(source)}$$

Model components

- $P(source|target)$ = translation model
- $P(target)$ = language model
- $P(source)$ = ignored because constant for an input

Translation model

How to compute $P(source|target) = P(s_1, \dots, s_n|t_1, \dots, t_n)$?

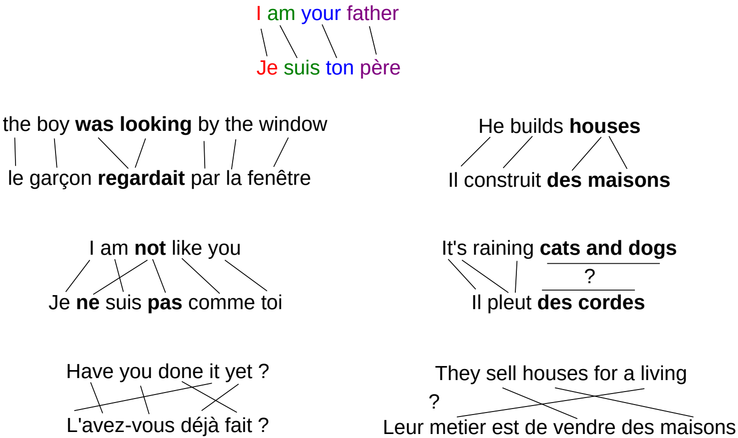
$$P(s_1, \dots, s_n|t_1, \dots, t_n) = \frac{nb(s_1, \dots, s_n \to t_1, \dots, t_n)}{\sum_x nb(x \to t_1, \dots, t_n)}$$

- Piecewise translation

$$\begin{aligned} P(\text{I am your father} \to \text{Je suis ton père}) = & P(\text{I} \to \text{je}) \times P(\text{am} \to \text{suis}) \\ & \times P(\text{your} \to \text{ton}) \\ & \times P(\text{father} \to \text{père}) \end{aligned}$$

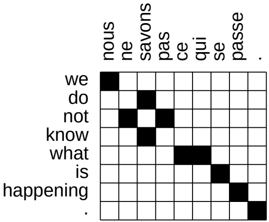
- To compute those probabilities
  - Need for alignment between source and target words

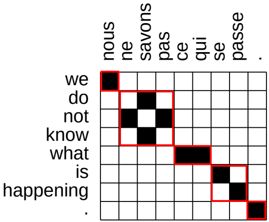
Alignements

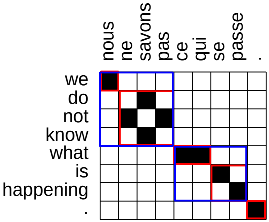


- Use bi-texts and alignment algorithm (such as Giza++)

Phrase table







"Phrase table"

we > nous
do not know > ne savons pas
what > ce qui
is happening > se passe
we do not know > nous ne savons pas
what is happening > ce qui se passe

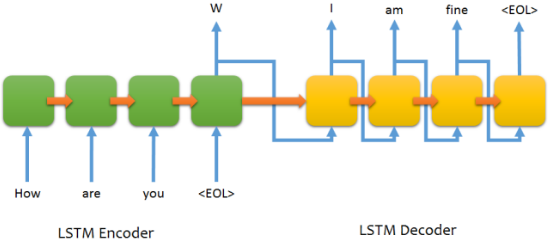
- Compute translation probability for all known phrases (an extension of n-gram language models)
  - Combine with LM and find best translation with decoding algorithm

Neural machine translation (NMT)

- Phrase-based translation
  - Same coverage problem as with word-ngrams
  - Alignment still wrong in 30% of cases
  - A lot of tricks to make it work
  - Researchers have progressively introduced NN
    - Language model
    - Phrase translation probability estimation
  - The google translate approach until mid-2016
- End-to-end approach to machine translation
  - Can we directly input source words and generate target words?

Encoder-decoder framework

- Generalisation of the conditioned language model
  - Build a representation, then generate sentence
  - Also called the seq2seq framework

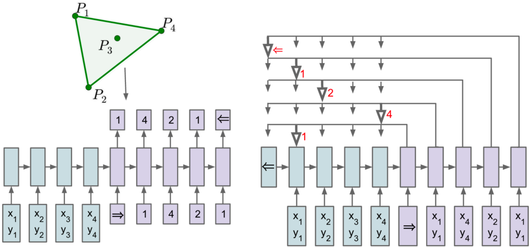


- But still limited for translation
  - Bad for long sentences
  - How to account for unknown words?
  - How to make use of alignments?

Interlude: Pointer networks

- Decision is an offset in the input
  - Number of classes dependent on the length of the input
  - Decision depends on hidden state in input and hidden state in output

$$y_i = softmax(v^\intercal tanh(Wr_j + Uh_i))$$



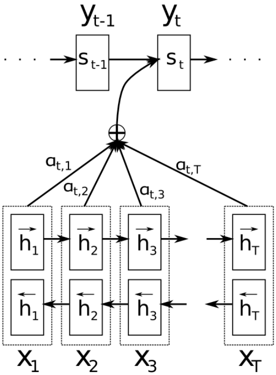
(a) Sequence-to-Sequence (b) Ptr-Net

Oriol Vinyals, Meire Fortunato, Navdeep Jaitly, "Pointer Networks", arXiv:1506.03134

Attention mechanisms

- Loosely based on human visual attention mechanism
  - Let neural network focus on aspects of the input to make its decision
  - Learn what to attend based on what it has produced so far

$$\alpha_i = \text{softmax}_j(f_{\text{align}}(d_i, e_j))$$
$$\text{attn}_i = \sum_j \alpha_{i,j} e_j$$
$$y_i = \text{softmax}(W\,[\text{attn}_i \oplus d_i] + b)$$



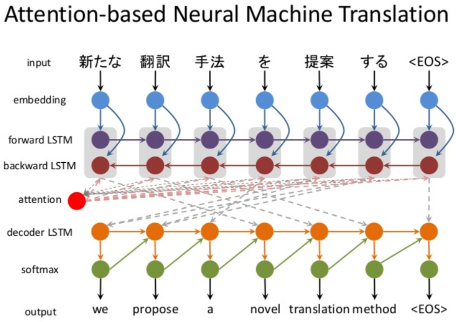
- Additive attention

$$f^{+}_{\text{align}}(d_i, e_j) = v^{\intercal} tanh(W_1 d_i + W_2 e_j)$$

- Multiplicative attention

$$f^{\times}_{\text{align}}(d_i, e_j) = d_i^{\intercal} W_3 e_j$$

Machine translation with attention



- Learns the word-to-word alignment

How to deal with unknown words

- If you don't have attention
  - Introduce unk symbols for low frequency words
  - Realign them to the input *a posteriori*
  - Use large translation dictionary or copy if proper name
- Use attention MT, extract α as alignment parameter
  - Then translate input word directly
- What about morphologically rich languages?
  - Reduce vocabulary size by translating word factors
    - Byte pair encoding algorithm
  - Use word-level RNN to transliterate word

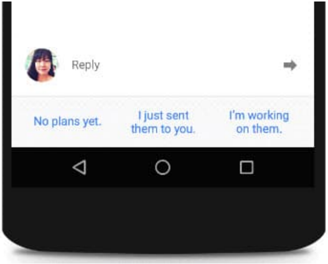
Zero-shot machine translation

- How to deal with the quadratic need for parallel data?
  - n languages $\rightarrow n^2$ pairs
  - So far, people have been using a pivot language ($x \rightarrow \text{english} \rightarrow y$)
- Parameter sharing across language pairs
  - Many to one $\rightarrow$ share the target weights
  - One to many $\rightarrow$ share the source weights
  - Many to many $\rightarrow$ train single system for all pairs
- Zero-shot learning
  - Use token to identify target language (ex: `<to-french>`)
  - Let model learn to recognize source language
  - Can process pairs never seen in training!
  - The model learns the "interlingua"
  - Can also handle code switching

"Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation", Johnson et al., arXiv:1611.04558

Conversation as translation

- Can we translate a question to its answer?
  - “Hello, how are you?” → “I am fine, thank you.”
  - “What is the largest planet in the solar system?” → “It is Jupiter.”
- “A Neural Conversational Model”, Vinyals et al, 2015
  - Train a seq2seq model to generate the next turn in a dialog
  - Led to the “auto answer” feature in Google Inbox

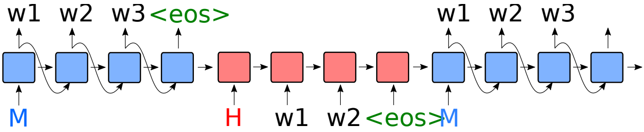


What is a chatbot?

- Dialog system which can have an entertaining conversation
  - Chit-chat
  - Task oriented
- History
  - Eliza, virtual therapist
    - http://www.masswerk.at/elizabot/
  - Mitsuku (best chatbot at Loebner price 2013/2016)
    - http://www.mitsuku.com/
  - The Microsoft Tay fiasco
    - Humans will always try to defeat an IA
  - A new industry hype
    - Facebook, google...
- Question: can we spare dialog model engineering?
  - Train a model directly from conversation traces

Chatbot 1: alternating language model

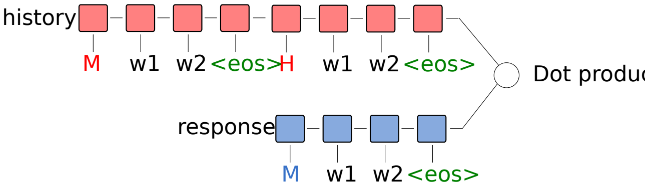
- A simplified version of the encoder-decoder (or seq2seq) framework
  - Trained the same way as a regular word-based language model
  - At prediction time, alternate between user input and generation
    - Training data needs to be in the same form




```
Human: my name is david . what is my name ?
Machine: david .
Human: my name is john . what is my name ?
Machine: john .
Human: are you a leader or a follower ?
Machine: i 'm a leader .
Human: are you a follower or a leader ?
Machine: i 'm a leader .
```

Chatbot 2: bi-encoder

- Learn a model that gives the same representation to an answer and the context that led to it
  - Information retrieval which can retrieve the next turn given a history
  - Encode history with a first recurrent model
  - Encode next turn with a second recurrent model
  - Compute a similarity between those representations (dot product)
- Training objective
  - Make sure the correct association has a higher score than a randomly selected pair
- Problem: the cost of retrieving a turn
  - Everything can be precomputed, just the dot product remains
  - Many approaches for finding approximate nearest neighbors in a high dimensional space (ie. locality preserving hashing)

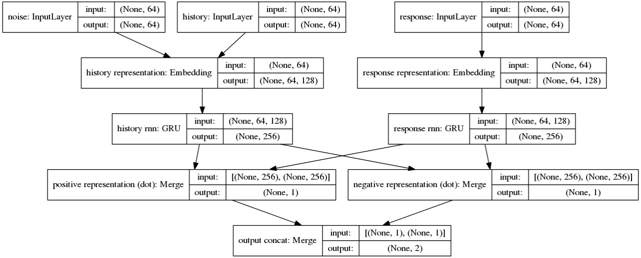


Bi-encoder training

- Maximize margin between the result of $h_i \cdot r_i$ and $n_i \cdot r_i$
  - h_i is the history
  - n_i is a random history
  - r_i is the response

$$Loss = \frac{1}{n} \sum_i \max(0, 1 - h_i \cdot r_i + n_i \cdot r_i))$$

- Keras model



Do we really need RNNs

- "Attention is all you need" [Vaswani et al, 2017]
  - Multiple layers of attention
- Position encoding
  - For position i, dimension j (total d, $k = 10000$)
  - $\mathrm{pe}_{i+k} = Linear(\mathrm{pe}_i)$

$$\mathrm{pe}_{i,2j} = sin(\frac{i}{k^{\frac{2j}{d}}})$$

$$\mathrm{pe}_{i,2j+1} = cos(\frac{i}{k^{\frac{2j}{d}}})$$

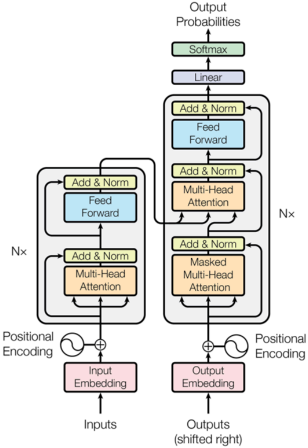
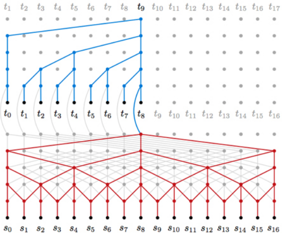


Figure 1: The Transformer - model architecture.

Explore other structures?

- WaveNet architecture
  - Extract long-term relations



- Account for parse tree
  - Generate annotations of the tree node

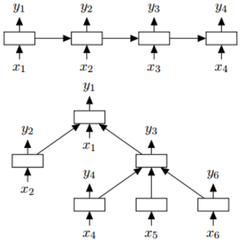
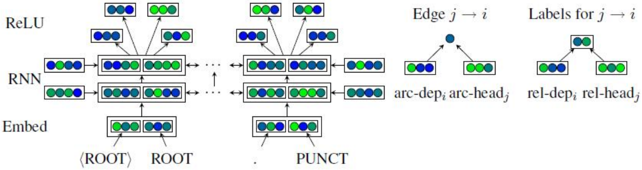


Figure 1: **Top:** A chain-structured LSTM network. **Bottom:** A tree-structured LSTM network with arbitrary branching factor.

Parsing

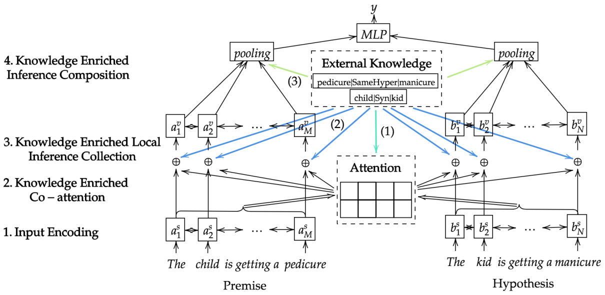
- Dependency parsing
  - Maximum spanning tree problem
- Deep bi-affine parser
  - "Deep Biaffine Attention for Neural Dependency Parsing", Dozat et al, 2016
  - Generate word representations with LSTMs, then combine hidden states to decide heads
    - ★ For head of word i:

$$y_i = softmax(h^{(head)} W h_i^{(dep)} + H^{(head)} b^\top)$$



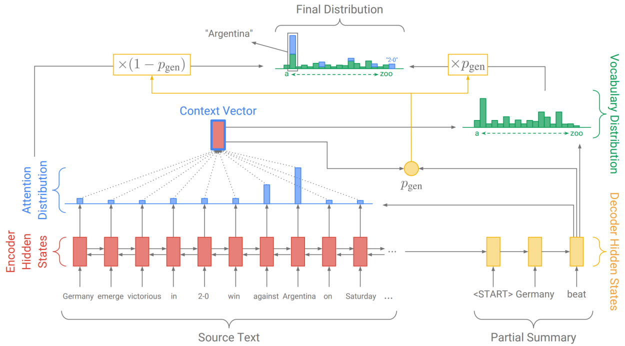
Textual inference

- Problem setup
  - Input: Hypothesis, Premise
  - Output: 3 categories: entailment, neutral, contradiction
- "Natural Language Inference with External Knowledge", (Chen et al, 2017)
  - $H \rightarrow P$ and $P \rightarrow H$ attention, followed by pooling
  - Bias attention with linguistic features from wordnet



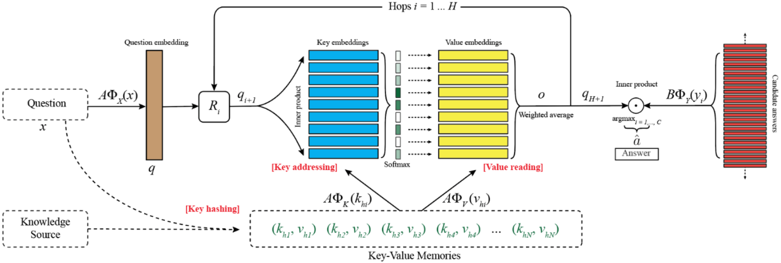
Summarization

- Automatic summarization
  - Given input text, generate short summary
- "Get To The Point: Summarization with Pointer-Generator Networks", (Abigail et al, ACL 2017)
  - When generating each word, decide between
    - ★ Copy from input (pointer network)
    - ★ Generate new word (language model)
  - Additional coverage penalty in loss
    - ★ Coverage factor in attention (sum of attention so far)



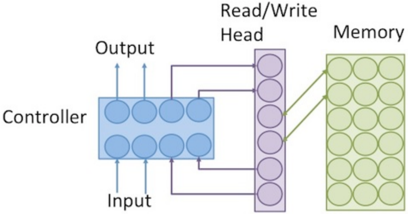
Memory networks

- Limitations of RNNs
  - Rewrite their memory at every time step
  - They have a fixed size memory
  - They need to reuse the same location in memory to perform the same action
- What if we had better memory devices
  - Static memory: Memory Networks (Weston et al., 2014)
    - ★ Memory containing representations (learned as part of the model)
    - ★ The model can do multiple passes over the memory to "deduce" its output



Neural Turing Machines

- Dynamic memory: Neural Turing Machines
  - At each round
    - Get memory read address from previous round
    - Combine input, state and memory into new memory
    - Generate memory read address for next round
  - Can learn basic algorithms
    - Copy, sort...

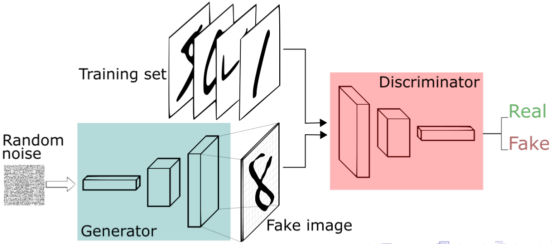


Generative Adversarial Networks

- Learn to generate samples starting from noise
  - minimize generator error $G(z) \rightarrow$ generate data from noise
  - maximize discriminator error $D(x) \rightarrow$ discriminate between noise and data

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

- While not converged:
  - for k steps:
    - ★ sample noise samples z
    - ★ sample training data samples x
    - ★ update discriminator to tell which is which
  - sample noise samples z
  - update generator to make good looking samples



Conclusion

- Add more prediction power to RNNs
  - Stacking
  - Bidirectional
  - Multitask
- Make better use of the input
  - Attention mechanisms
- Fancy applications
  - Machine translation
  - Caption generation
  - Chatbots
  - ...

Remaining challenges

Deep learning for NLP

- Language independence
  - We still need training data in all languages
- Domain adaptation
  - Often, we have plenty of data where we don't need it, and none where we would need it
  - What if the test data does not follow the distribution of training data?
- Dealing with small datasets
  - Annotating complex phenomena is expensive

Deep learning

- Efficient training on CPU, mobile devices
  - Binary neural networks
- Training non differentiable systems
  - Reinforcement learning
- Reasoning, world knowledge...
  - AI, here we are