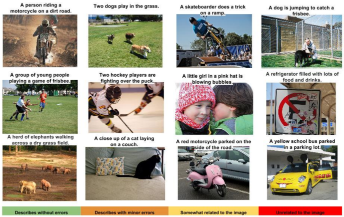
Deep Learning

Applications of RNNs

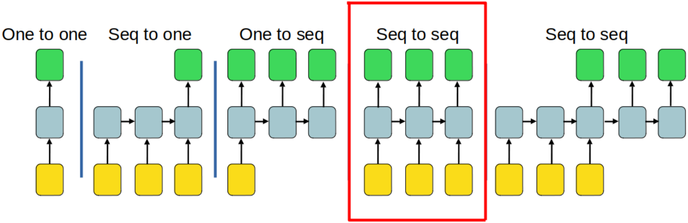

A person riding a motorcycle on a dirt road.
Two dogs play in the grass.
A skateboarder does a trick on a ramp.
A dog is jumping to catch a frisbee.
A group of young people playing a game of frisbee.
Two hockey players are fighting over the puck.
A little girl in a pink hat is blowing bubbles.
A refrigerator filled with lots of food and drinks.
A herd of elephants walking across a dry grass field.
A close up of a cat laying on a couch.
A red motorcycle parked on the side of the road.
A yellow school bus parked in a parking lot.
Describes without errors
Describes with minor errors
Somewhat related to the image
Unrelated to the image

Learning goals

- Understand application to Language Modelling
- Get to know Encoder-Decoder architectures
- Learn about further RNN applications

Language Modelling

Seq-to-Seq (Type I)



RNNS - LANGUAGE MODELLING

- In an earlier example, we built a ’sequence-to-one’ RNN model to perform ’sentiment analysis’.
- Another common task in Natural Language Processing (NLP) is **’language modelling’**.
- Input: word/character, encoded as a one-hot vector.
- Output: probability distribution over words/characters given previous words

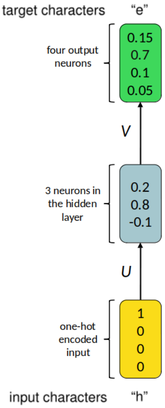
$$\mathbb{P}(y^{[1]}, \ldots, y^{[T]}) = \prod_{i=1}^{T} \mathbb{P}(y^{[i]} | y^{[1]}, \ldots, y^{[i-1]})$$

→ given a sequence of previous characters, ask the RNN to model the probability distribution of the next character in the sequence!

RNNS - LANGUAGE MODELLING

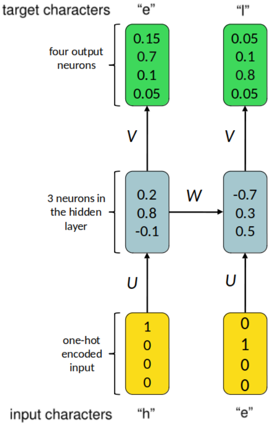
- In this example, we will feed the characters in the word "hello" one at a time to a ’seq-to-seq’ RNN.
- For the sake of the visualization, the characters "h", "e", "l" and "o" are one-hot coded as a vectors of length 4 and the output layer only has 4 neurons, one for each character (we ignore the <eos> token).
- At each time step, the RNN has to output a probability distribution (softmax) over the 4 possible characters that might follow the current input.
- Naturally, if the RNN has been trained on words in the English language:
  - The probability of “e” should be likely, given the context of “h”.
  - “l” should be likely in the context of “he”.
  - “l” should **also** be likely, given the context of “hel”.
  - and, finally, “o” should be likely, given the context of “hell”.

RNNS - LANGUAGE MODELLING



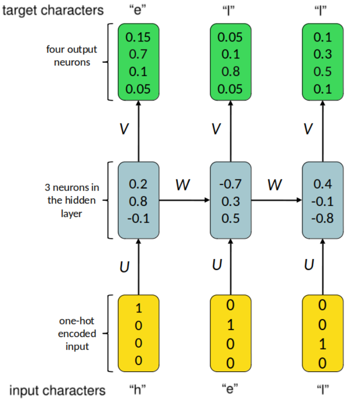
The probability of “e” should be high, given the context of “h”.

RNNS - LANGUAGE MODELLING



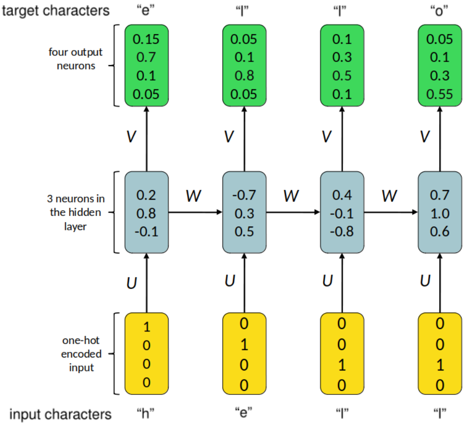
The probability of “l” should be high, given in the context of “he”.

RNNS - LANGUAGE MODELLING



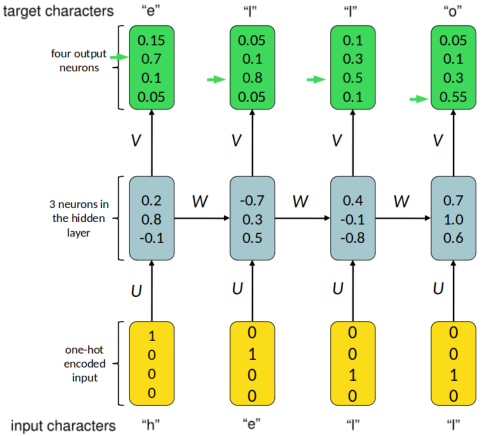
The probability of “l” should **also** be high, given in the context of “hel”.

RNNS - LANGUAGE MODELLING



The probability of "o" should be high, given the context of "hell".

RNNS - LANGUAGE MODELLING



During training, our goal would be to increase the confidence for the correct letters (indicated by the green arrows) and decrease the confidence of all others.

WORD EMBEDDINGS

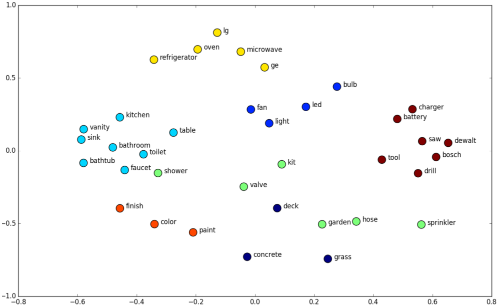


Figure: Two-dimensional embedding space. Typically, the embedding space is much higher dimensional (Ruizendaal, 2018).

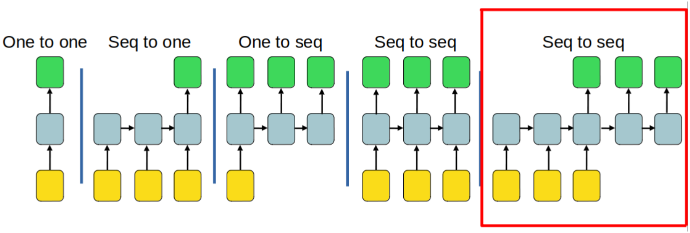
- Instead of one-hot representations of words it is standard practice to encode each word as a dense (as opposed to sparse) vector of fixed size that captures its underlying semantic content.
- Similar words are embedded close to each other in a lower-dimensional embedding space.

WORD EMBEDDINGS

- The dimensionality of these embeddings is typically much smaller than the number of words in the dictionary.
- Using them gives you a "warm start" for any NLP task. It is an easy way to incorporate prior knowledge into your model and a rudimentary form of **transfer learning**.
- Two very popular approaches to learn word embeddings are **word2vec** by Google and **GloVe** by Facebook. These embeddings are typically 100 to 1000 dimensional.
- Even though these embeddings capture the meaning of each word to an extent, they do not capture the *semantics* of the word in a given context because each word has a static precomputed representation. For example, depending on the context, the word "bank" might refer to a financial institution or to a river bank.

Encoder-Decoder Architectures

Seq-to-Seq (Type II)



ENCODER-DECODER NETWORK

- For many interesting applications such as question answering, dialogue systems, or machine translation, the network needs to map an input sequence to an output sequence of different length.
- This is what an encoder-decoder (also called sequence-to-sequence architecture) enables us to do!

ENCODER-DECODER NETWORK

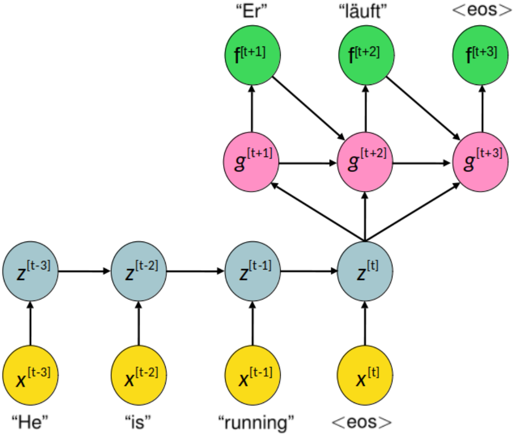


Figure: In the first part of the network, information from the input is encoded in the context vector, here the final hidden state, which is then passed on to every hidden state of the decoder, which produces the target sequence.

ENCODER-DECODER NETWORK

- An input/encoder-RNN processes the input sequence of length n_x and computes a fixed-length context vector C, usually the final hidden state or simple function of the hidden states.
- One time step after the other information from the input sequence is processed, added to the hidden state and passed forward in time through the recurrent connections between hidden states in the encoder.
- The context vector summarizes important information from the input sequence, e.g. the intent of a question in a question answering task or the meaning of a text in the case of machine translation.
- The decoder RNN uses this information to predict the output, a sequence of length n_y, which could vary from n_x.

ENCODER-DECODER NETWORK

- In machine translation, the decoder is a language model with recurrent connections between the output at one time step and the hidden state at the next time step as well as recurrent connections between the hidden states:

$$\mathbb{P}(y^{[1]}, \ldots, y^{[n_y]} | \mathbf{x}^{[1]}, \ldots, \mathbf{x}^{[n_x]}) = \prod_{t=1}^{n_y} p(y^{[t]} | C; y^{[1]}, \ldots, y^{[t-1]})$$

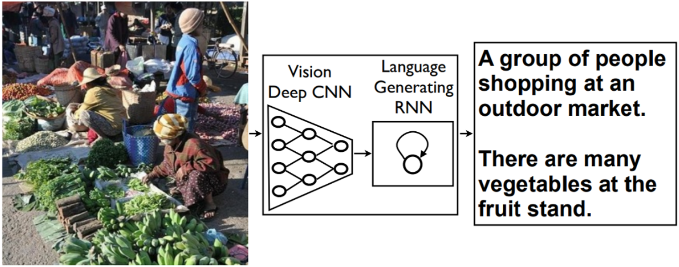
  with C being the context-vector.
- This architecture is now jointly trained to minimize the translation error given a source sentence.
- Each conditional probability is then

$$p(y^{[t]} | y^{[1]}, \ldots, y^{[t-1]}; C) = f(y^{[t-1]}, g^{[t]}, C)$$

  where f is a non-linear function, e.g. the tanh and $g^{[t]}$ is the hidden state of the decoder network.

More application examples

SOME MORE SOPHISTICATED APPLICATIONS



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

Figure: Show and Tell: A Neural Image Caption Generator. A language generating RNN tries to describe in brief the content of different images (Vinyals et al., 2014).

SOME MORE SOPHISTICATED APPLICATIONS

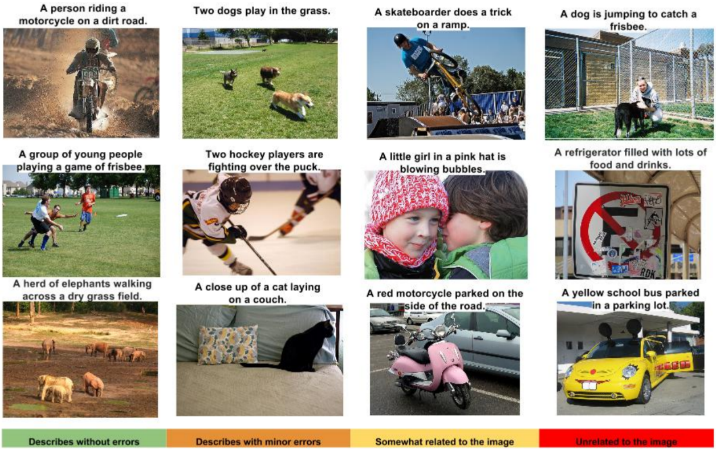

A person riding a motorcycle on a dirt road.
Two dogs play in the grass.
A skateboarder does a trick on a ramp.
A dog is jumping to catch a frisbee.
A group of young people playing a game of frisbee.
Two hockey players are fighting over the puck.
A little girl in a pink hat is blowing bubbles.
A refrigerator filled with lots of food and drinks.
A herd of elephants walking across a dry grass field.
A close up of a cat laying on a couch.
A red motorcycle parked on the side of the road.
A yellow school bus parked in a parking lot.
Describes without errors
Describes with minor errors
Somewhat related to the image
Unrelated to the image

Figure: Show and Tell: A Neural Image Caption Generator. A language generating RNN tries to describe in brief the content of different images (Vinyals et al., 2014).

SOME MORE SOPHISTICATED APPLICATIONS

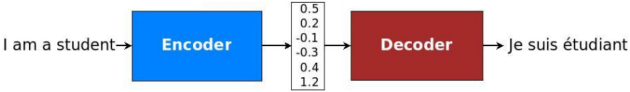


Figure: Neural Machine Translation (seq2seq): Sequence to Sequence Learning with Neural Networks. An encoder converts a source sentence into a “meaning” vector which is passed through a decoder to produce a translation (Sutskever et al., 2014).

SOME MORE SOPHISTICATED APPLICATIONS

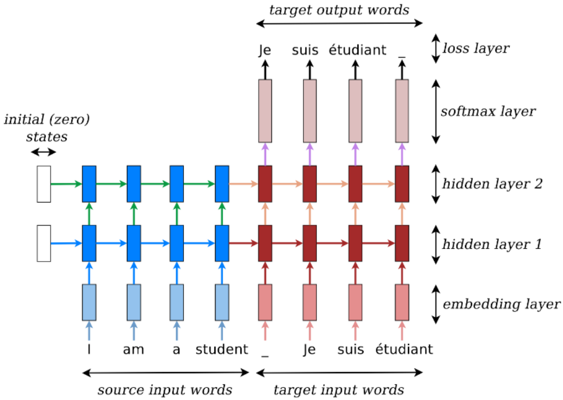


Figure: Neural Machine Translation (seq2seq): Sequence to Sequence Learning with Neural Networks. An encoder converts a source sentence into a "meaning" vector which is passed through a decoder to produce a translation (Sutskever et al., 2014).

SOME MORE SOPHISTICATED APPLICATIONS

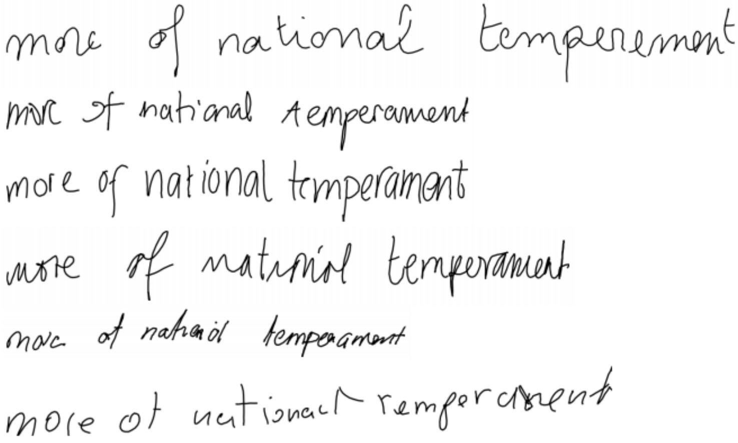

more of national temperement
more of national temperament
more of national temperamant
more of natinil temperament
more of nationil temperament
more ot nationact temperarent

Figure: Generating Sequences With Recurrent Neural Networks. Top row are real data, the rest are generated by various RNNs (Graves, 2014).

SOME MORE SOPHISTICATED APPLICATIONS

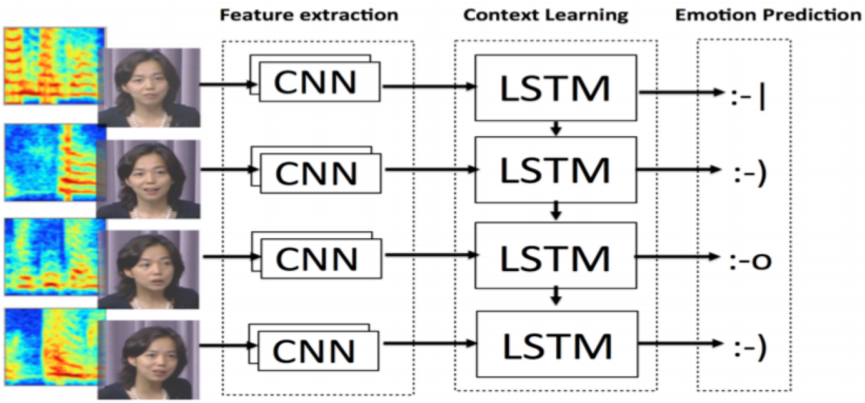


Figure: Convolutional and recurrent nets for detecting emotion from audio data (Anand, 2015).

SOME MORE SOPHISTICATED APPLICATIONS

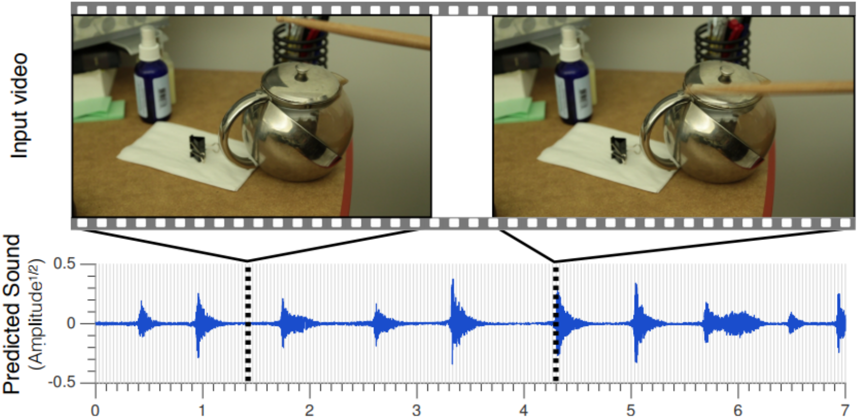


Figure: Visually Indicated Sounds. A model to synthesize plausible impact sounds from silent videos. ▸ Click here (Owens et al., 2016)

REFERENCES

- Ruizendaal, R. (2018, October 21). *Using deep learning for structured data with entity embeddings*. Medium. `https://towardsdatascience.com/deep-learning-structured-data-8d6a278f3088`

- Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). *Show and Tell: A Neural Image Caption Generator.*

- Graves, A. (2014). Generating Sequences With Recurrent Neural Networks.

- Anand, N. (2015). *Convoluted Feelings Convolutional and recurrent nets for detecting emotion from audio data.* `https://api.semanticscholar.org/CorpusID:209374156`

- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks.*

- Owens, A., Isola, P., McDermott, J., Torralba, A., Adelson, E. H., & Freeman, W. T. (2016). *Visually Indicated Sounds*.