UNIVERSITÄT
ZU KÖLN

DEEP LEARNING – SESSION 11

WiSe 2024/2025

Janis Pagel

Institut für Digital Humanities 2025-01-09

01

SOLUTION EXERCISE 10

Discussion Exercise 10

- Solution at `https://github.com/IDH-Cologne-Deep-Learning-2024/Exercise-10/blob/main/lstm_solution.py`

Recap

- So far, we have only looked at neural networks (mostly) as classifiers
  - Classes to classify as output, embeddings as input
- But: Current state-of-the-art deep learning uses language modelling
  - For classification
  - For text generation
  - For everything else

02

LANGUAGE MODELLING

Language Modelling

- Model that outputs probabilities about the likelihood that a word follows a given sequence of words
- Example (probabilities made up):
  - "The capital of Germany is"
    - "Berlin" (70%)
    - "Bonn" (15%)
    - ...
    - "Paris" (1%)
    - "nice" (0.5%)
    - ...
    - "is" (0.00000001%)
    - "of" (0.000000001%)
    - ...
- All probabilities for the vocabulary words need to add up to 100%
- Probabilities are learned on large collections of texts

Formalization of Language Models

- Every sequence of words receives probability:

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i|w_{1:i-1}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_n|w_{1:n-1})$$

- This means LMs can be used as both
  - *Analyzers*
    - Calculate the probability of a given sequence (i.e. "How likely is it that this sequence will occur")
  - *Generators*
    - What is the probability of all possibly next words, pick the most likely (or from a collection of the most likely)
- In practice it is often too costly (time and resource-wise) to calculate the probability based on all sequences ever occurring before
  - Assumption that the next word only depends on the previous k words

$$P(w_{n+1}|w_{1:n}) \approx P(w_{n+1}|w_{n-k:n})$$

  - This is not necessary for neural network-based language models like RNNs/LSTMs

Perplexity

- Perplexity is a measure that can be used to see how good a LM predicts the sequences in a given corpus
  - If perplexity is low, the LM predicts the given corpus well (assigns high probabilities to the sequences in the corpus)
  - If the perplexity is high, the LM does not predict the given corpus well (assigns low probabilities to the sequences in the corpus)
- Perplexity can be calculated as

$$perplexity(w_{1:n}) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i|w_{1:i-1})}}$$

- Perplexity is corpus-dependent
  - A LM that works well on one corpus might not work well for another corpus
  - Perplexity of different LMs can only be compared on the same corpus

RNNs/LSTMs as language models

- In deep learning, RNNs/LSTMs can easily be used as language models
- The input is a sequence of word embeddings
- The output is a probability distribution over all possibly occurring next words (softmax)

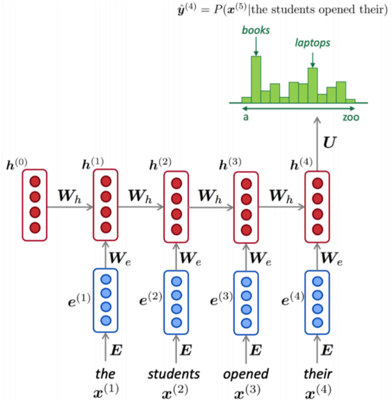
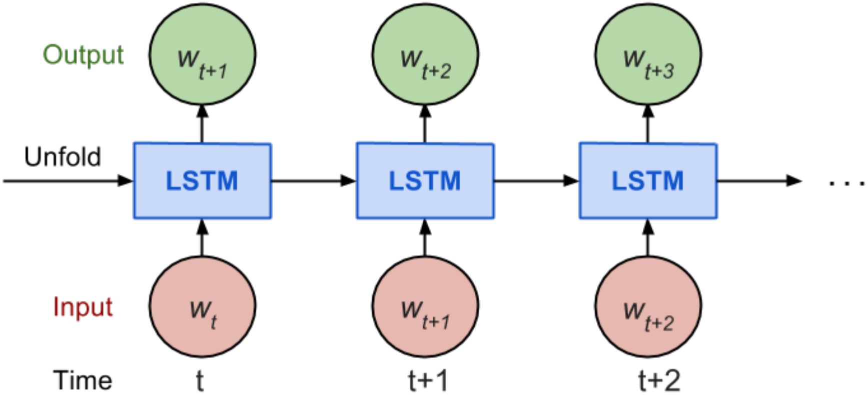


Figure: Source: https://pantelis.github.io/cs634/docs/common/lectures/nlp/rnn-language-models/

RNNs/LSTMs as language models

- The model simply predicts the next word at every time step



Encoder-Decoder

- Usually, the generation with a LM is unconditional
  - The LM will just produce text based on the probability distribution it learned, without an end or goal
- When the generation should be conditional (for example translation, chat bot, summarization, etc.), an *encoder-decoder* setup can be used
  - The encoder *encodes* the input into a latent representation
  - The decoder *decodes* the latent representation into the desired output

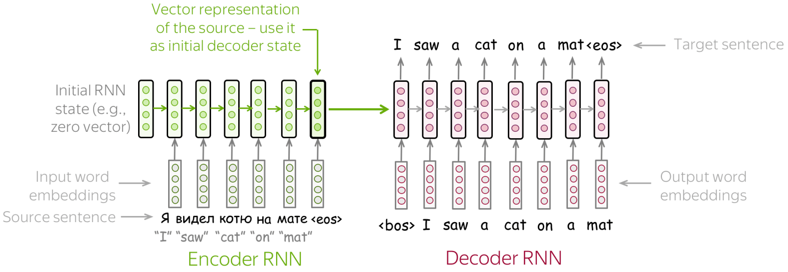


Figure: Source: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention

- In the encoder-decoder setup, the encoder outputs a single vector that gets interpreted by the decoder
  - This vector contains all information about the input sequence in a compressed form
  - The decoder needs to interpret the input based on this compressed representation alone
- Better: Let the model learn what part of the encoder input is most relevant for the decoder
  - Similar to the LSTM mechanism
- This is called *attention*
  - Simply another representation (vectors) learned together with the encoder and decoder

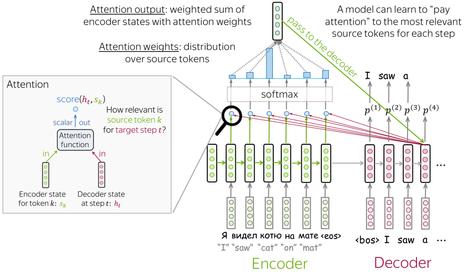


Figure: Source: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention

- It was shown early on that attention helps to connect related tokens in the mapping sequences

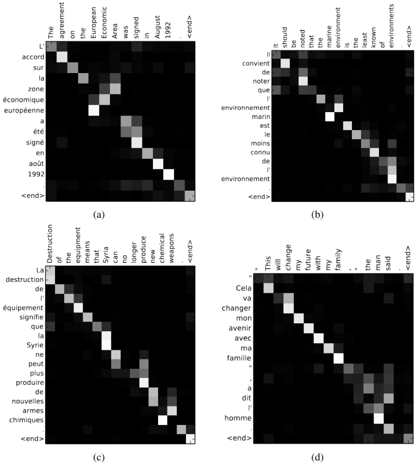


Figure: Source: Bahdanau, Cho, and Bengio (2014, p. 6)

03

LANGUAGE MODELS IN KERAS

Language Model in Keras

- A language model in Keras is like a usual model with X being a sequence and y being the next word

```
model = Sequential()
model.add(Embedding(vocab_size, 300, input_length=maxlen))
model.add(LSTM(64))
model.add(Dense(vocab_size, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.fit(X, y)
```

Encoder-Decoder in Keras

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input, Embedding, LSTM, RepeatVector
model = Sequential()
model.add(Input(shape=(INPUT_LENGTH,)))
model.add(Embedding(input_dim = number_of_symbols, output_dim =64,))
model.add(LSTM(64, return_sequences = False)) # Encoder
model.add(RepeatVector(OUTPUT_LENGTH))
model.add(LSTM(64, return_sequences=True, dropout=0.2)) # Decoder
model.add(Dense(number_of_symbols*2, activation='softmax')
```

04

EXERCISE 11

Exercise 11

- Exercise 11 can be found on https://github.com/IDH-Cologne-Deep-Learning-2024/Exercise-11
- Deadline: January 09, 2025, 08:00:00 CET

UNIVERSITY
OF COLOGNE

Janis Pagel
Institut für Digital Humanities

eMail janis.pagel@uni-koeln.de
Homepage https://janispagel.de
Phone +49 221 470 5749

Foto: Gregor Hübl

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. Version 7. DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473 [cs.CL]. URL: https://arxiv.org/abs/1409.0473.