Deep Learning

Übung WS 23/24

Judith Nester (nester@uni-koeln.de)

18-01-2024

Recap

- » Sequential Text Data
  - Text data organized in a chronological order
  - Information unfolds over time in a structured manner.
- » RNN (Recurrent Neural Network)
  - Utilizes recurrent connections to capture temporal dependencies
  - Each network layer processes input while maintaining a hidden state
  - Well-suited for applications where context or order of input matters
  - Very slow, looses dependencies over long distances

Recap

- LSTM (Long Short-Term Memory)
  - »Extended neurons« that can also store a state over long distances
  - Allows modeling (and learning) long distance dependencies in language

Recap

- » LSTM (Long Short-Term Memory)
    - »Extended neurons« that can also store a state over long distances
    - Allows modeling (and learning) long distance dependencies in language
- » Problems with LSTM
    - Very slow (even slower than basic RNNs)
    - Context that's far away from the current timestep gets lost
- » Also: What happens with sequence-to-sequence tasks when sequences are not equally long?
    - E.g.: »I don't know« $\rightarrow$ »Ich weiß nicht«
    - E.g.: »111101001010« $\rightarrow$ »3914«
    - Today: Encoder-Decoder-Networks

Today

Section 1

Encoder-Decoder

Introduction

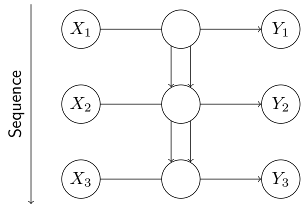


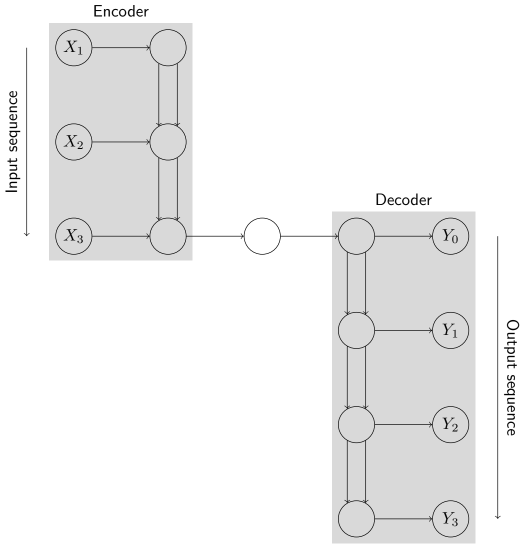
Figure: Neural Network with an LSTM Layer

- Each X value leads to a Y value
- Network has no way to skip a sequence element
- Many real world sequence labeling tasks are n-to-m-tasks
  - n elements in one sequence are associated with m element in the other

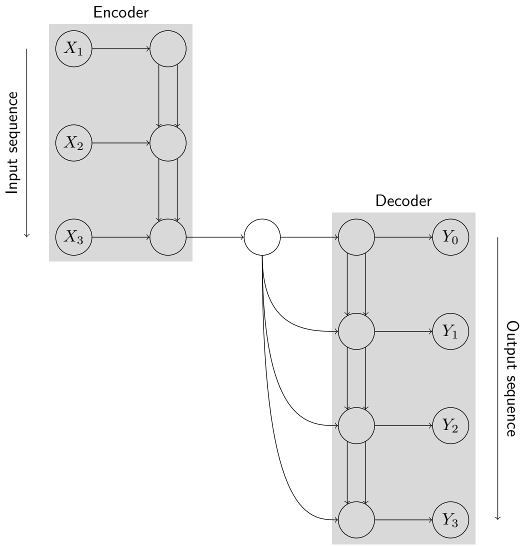
Encoder-Decoder-Architecture

- Network has two parts:
  - Encoder maps from input data to an internal representation
  - Decoder maps from internal representation to the output
- Internal representation
  - Use the output or internal state of last cell
  - Not interpretable

Encoder-Decoder-Architecture



Encoder-Decoder-Architecture



Encoder-Decoder-Architecture in Keras

- Encoder
  - Regular input layer
  - LSTM-Layer with `return_sequences=False`
    - Because we don't want a sequence as output, but just the output of the last cell
- Decoder
  - Every output sequence element gets the ›internal representation‹ as input
    - Thus, it needs to be repeated with the `RepeatVector()` layer
  - LSTM-Layer with `return_sequences=True`
    - Because now, we want the sequence
  - Output layer as before
    - With one-hot-encoding for multi-class problems

Encoder-Decoder-Architecture in Keras

The Code

```
model = models.Sequential()
model.add(layers.Input(shape=(INPUT_LENGTH,)))
model.add(layers.Embedding(input_dim=number_of_symbols, output_dim=64,))
model.add(layers.LSTM(64, return_sequences=False))
model.add(layers.RepeatVector(OUTPUT_LENGTH))
model.add(layers.LSTM(32, return_sequences=True, dropout=0.2))
model.add(layers.Dense(number_of_symbols*2, activation='softmax'))
```

Encoder-Decoder Architecture

- Translation Tasks:
  - Often used in machine translation, text-to-speech, and image-to-text applications.
- Advantages:
  - Flexibility for various tasks.
  - Ability to learn complex mappings between input and output.
- Challenges:
  - Limited processing of long sequences.
  - Difficulties in maintaining long-term dependencies.
  - Limited parallelization options, leading to slower training times.

Section 2

Transformer

Transformer - A Revolutionary Encoder-Decoder Architecture

» Specific architecture for sequential data, introduced in "Attention is all you need" by Vaswani et al. in 2017
» Utilizes self-attention mechanisms for effective sequence processing
» Self-attention layers for parallelized processing.

Benefits of Transformers

- Transformers do not process sequences in time steps, but perform processing in parallel
- They are fast
- Division in Encoder block and Decoder block makes Transformer architecture flexible
- Can be separated and rearranged to solve more than just translation tasks
- Good scalability for larger datasets and models

Applications

- Transformer structures find applications in various domains, e.g., computer vision
- Continued work on Transformer variants such as BERT, GPT, and their adaptations for specific tasks
- Inspires a new generation of models in AI research

Section 3

Attention

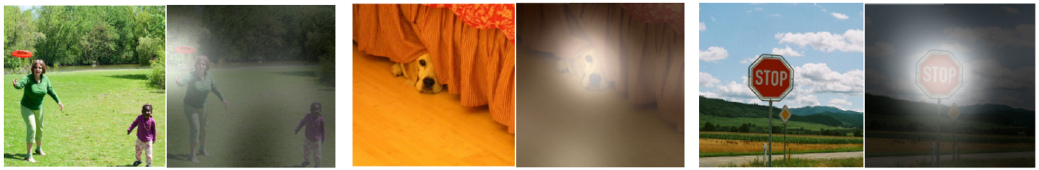

A woman is throwing a frisbee in a park.
A dog is standing on a hardwood floor.
A stop sign is on a road with a mountain in the background.

Figure: Examples of attending to the correct object (Xu et al., 2015)

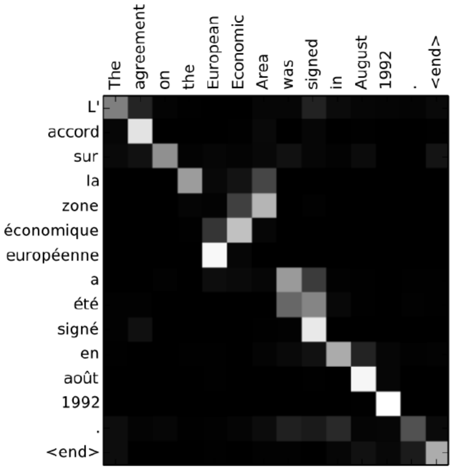


Figure: Attention paid by a neural machine translation network (Bahdanau et al., 2015)

Introduction

- A mechanism to allow the network to learn what to focus on
- Idea: Not all parts of the input are equally important
  - MT: »la zone économique européenne« $\rightarrow$ »the European Economic Area«, irrespective of context

Introduction

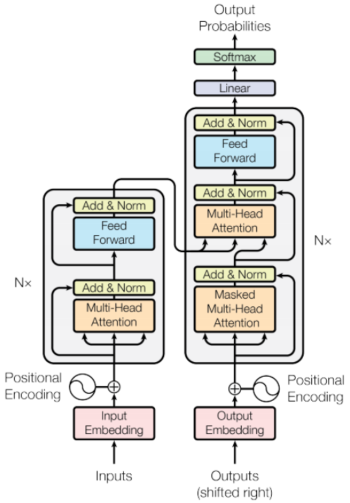
- A mechanism to allow the network to learn what to focus on
- Idea: Not all parts of the input are equally important
  - MT: »la zone économique européenne« $\rightarrow$ »the European Economic Area«, irrespective of context
- Mirrows human reading/translating activities
- Developed for machine translation, then applied to other tasks

Introduction

- A mechanism to allow the network to learn what to focus on
- Idea: Not all parts of the input are equally important
  - MT: »la zone économique européenne« → »the European Economic Area«, irrespective of context
- Mirrows human reading/translating activities
- Developed for machine translation, then applied to other tasks

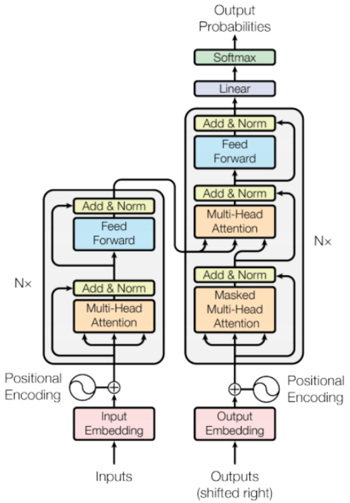
Dzmitry Bahdanau et al. (2015). »Neural Machine Translation by Jointly Learning to Align and Translate«. *In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. url: `http://arxiv.org/abs/1409.0473`*

Encoder-Decoder-Architecture



Encoder-Decoder-Architecture with Attention Layers

- Text in the first language is passed through the **Encoder block**
  - Input is converted into Embeddings and added with Positional Encodings
  - Attention vector for each word, representing the relevance of a word in relation to its context
- **Decoder block** uses two Attention layers and one Feedforward Network
  - Sentence in first language is mapped to a representation of the same meaning in the second language
  - Softmax function produces a probability distribution



Section 4

Transfer Learning

Introduction

- Typical scenario
  - Large data set of newspaper texts
  - Small data set of texts in your domain

Introduction

- Typical scenario
  - Large data set of newspaper texts
  - Small data set of texts in your domain
- Domain adaptation: Adapt your models from one domain to another
- Transfer learning: One way to do this

Transfer Learning

- Recipe
  - Define a model, train it on large data set
  - Freeze the layers by setting `trainable = False`
  - Define a new model on top of the model
  - Train on smaller data set

Transfer Learning

Components

- »Freeze the layers by setting `trainable = False`«
  - Training: Estimating weights of layers
  - Freezing: Disable weight updating
  - Some or all layers can be frozen
  - Not frozen layers are updated as usual

Transfer Learning

Components

- »Freeze the layers by setting `trainable = False`«
  - Training: Estimating weights of layers
  - Freezing: Disable weight updating
  - Some or all layers can be frozen
  - Not frozen layers are updated as usual
- »Define a new model on top of the model«
  - An entire model can be used similar to a layer!

Example

```
input = Input(shape=(12,))
emb = Embedding(output_dim = 150)(input)
hl1 = Dense(150)(emb)
hl2 = Dense(1)(hl1)

base_model = Model(inputs=input, outputs=hl2)
# train on large data set
base_model.fit(...)

# freeze
base_model.trainable = False

# define new model, using the base model as layer
input = Input(shape=(21,))
x = base_model(input, training=False)
x = Dense(2)(x)
outputs = Dense(1)(x)
model = Model(inputs=input, outputs=outputs)
```

Section 5

Hugging Face

Introduction

- An AI company that provides
  - A Python library for transformer models
    - Since 2.0 compatible with tensorflow/keras and PyTorch
  - A platform to share BERT models (e.g., for different languages) and/or data sets
  - Some paid services

Introduction

- An AI company that provides
  - A Python library for transformer models
    - Since 2.0 compatible with tensorflow/keras and PyTorch
  - A platform to share BERT models (e.g., for different languages) and/or data sets
  - Some paid services

Installation

```
pip install transformers
```

Code

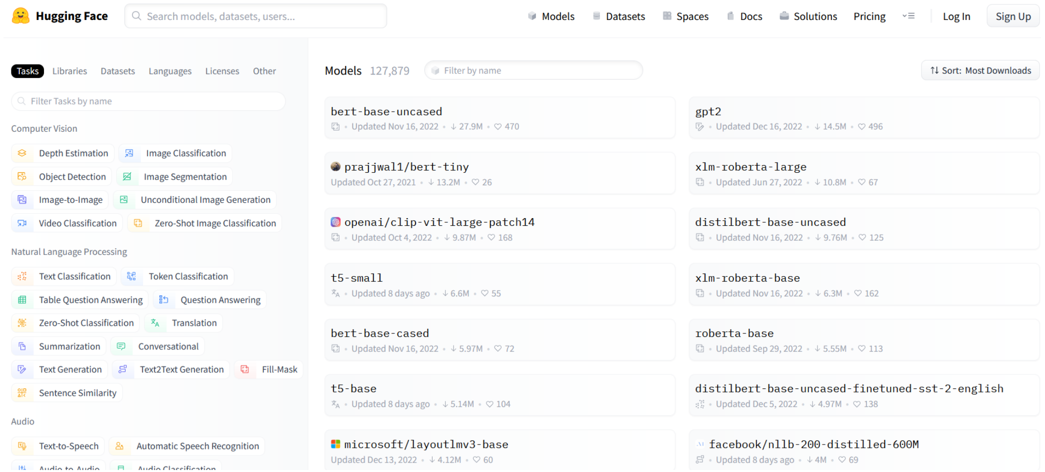
```python
import tensorflow as tf
from transformers import TFAutoModelForSequenceClassification

# Load model as keras model
model = TFAutoModelForSequenceClassification
.from_pretrained("bert-base-cased", num_labels=2)

# do the usual keras stuff
model.compile(...)

# fine-tuning
model.fit(...)
```

https://huggingface.co/transformers/training.html



Models 127,879

bert-base-uncased
Updated Nov 16, 2022 • ↓ 27.9M • ♡ 470

gpt2
Updated Dec 16, 2022 • ↓ 14.5M • ♡ 496

prajjwal1/bert-tiny
Updated Oct 27, 2021 • ↓ 13.2M • ♡ 26

xlm-roberta-large
Updated Jun 27, 2022 • ↓ 10.8M • ♡ 67

openai/clip-vit-large-patch14
Updated Oct 4, 2022 • ↓ 9.87M • ♡ 168

distilbert-base-uncased
Updated Nov 16, 2022 • ↓ 9.76M • ♡ 125

t5-small
Updated 8 days ago • ↓ 6.6M • ♡ 55

xlm-roberta-base
Updated Nov 16, 2022 • ↓ 6.3M • ♡ 162

bert-base-cased
Updated Nov 16, 2022 • ↓ 5.97M • ♡ 72

roberta-base
Updated Sep 29, 2022 • ↓ 5.55M • ♡ 113

t5-base
Updated 8 days ago • ↓ 5.14M • ♡ 104

distilbert-base-uncased-finetuned-sst-2-english
Updated Dec 5, 2022 • ↓ 4.97M • ♡ 138

microsoft/layoutlmv3-base
Updated Dec 13, 2022 • ↓ 4.12M • ♡ 60

facebook/nllb-200-distilled-600M
Updated 8 days ago • ↓ 4M • ♡ 69

Section 6

Exercise

Exercise 09

Take a look at the variety of Transformer models on Huggingface. Are there any models that you might be able to use for your module exam? What data do they use? What kind of tasks do they train on? Find the code and try to understand it. Next week we will look at a few Transformer models together.