Recap: Overfitting and Recurrent Neural Networks

- Overfitting
  - The model did not generalize well
  - Not purely technical problem
  - DL answers: regularization, dropout
- Recurrent Neural Networks
  - Basic neural networks: Classify one item at a time
  - RNN
    - Additional connection along the sequence
    - Information can be passed from one sequence element to the next
    - One dimension more, because training instance is a sequence

UNIVERSITÄT ZU KÖLN

Machine Learning: Transformer Models, BERT, The Future?

VL Sprachliche Informationsverarbeitung

Nils Reiter
nils.reiter@uni-koeln.de

January 18, 2024
Winter term 2023/24

INSTITUT FÜR DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Introduction

- (Recurrent) neural networks provide building blocks
- Powerful machine learning, usable for many different tasks
- RNN/Bi-LSTM have taken over NLP landscape – 2015–2018

Introduction

- (Recurrent) neural networks provide building blocks
- Powerful machine learning, usable for many different tasks
- RNN/Bi-LSTM have taken over NLP landscape – 2015–2018

Current State of the Art: Transformer architecture

- Encoder-Decoder-Network — Sutskever et al. (2014)
- Attention layer — Vaswani et al. (2017)
- New training paradigm(s)

Section 1

Encoder-Decoder-Networks

Introduction

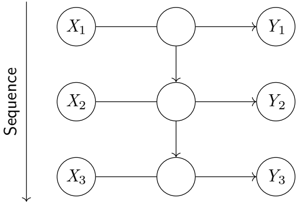


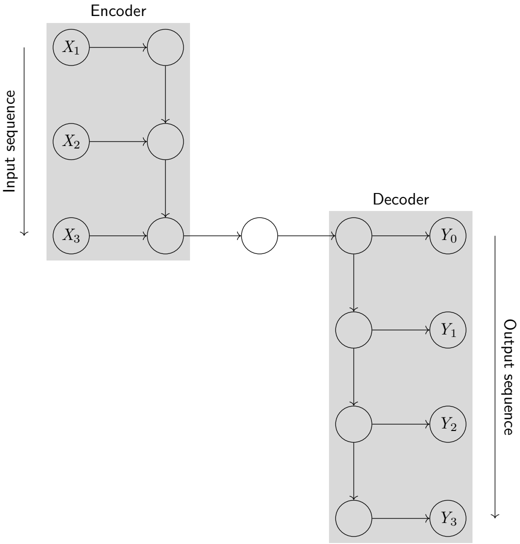
Figure: Neural network with a recurrent layer

- 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

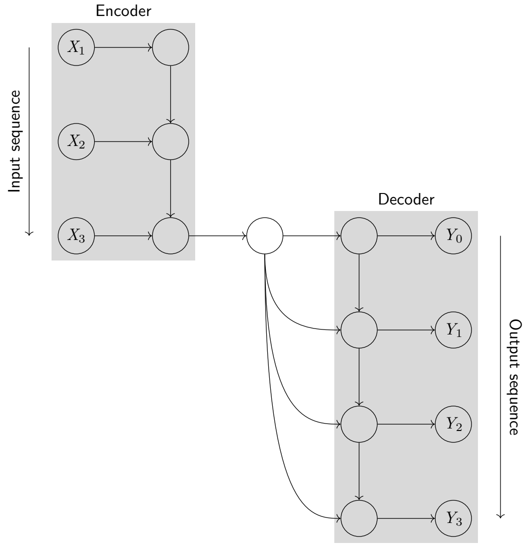
Encoder-Decoder-Architecture

- Network has two parts:
  - Encoder maps from input data to an internal representation
  - Internal representation optionally processed by a regular dense layer
  - Decoder maps from internal representation to the output
- Internal representation
  - Use the output of last recurrent neuron
    - Or internal state of last recurrent cell
  - Some vector, not interpretable

Encoder-Decoder-Architecture



Encoder-Decoder-Architecture



Encoder-Decoder-Architecture in Keras

- Encoder
  - Regular input layer
  - Recurrent 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
    - This is just copying the vector
  - Recurrent 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

Listing 1: The Code

```
model = models.Sequential()
# Encoder
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))

# Copy the internal representation (optional)
model.add(layers.RepeatVector(OUTPUT_LENGTH))

# Decoder
model.add(layers.LSTM(32, return_sequences=True))
model.add(layers.Dense(number_of_symbols*2, activation='softmax'))
```

Section 2

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)

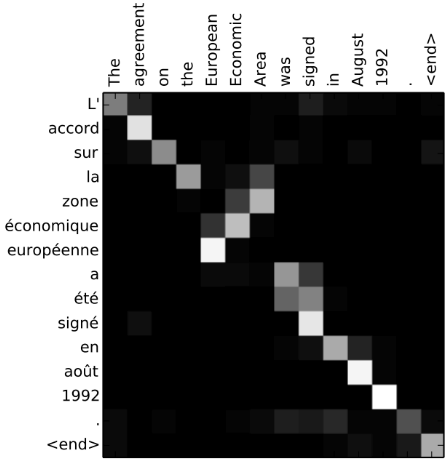


Figure: Attention paid by a neural machine translation network (Bahdanau et al., 2015)

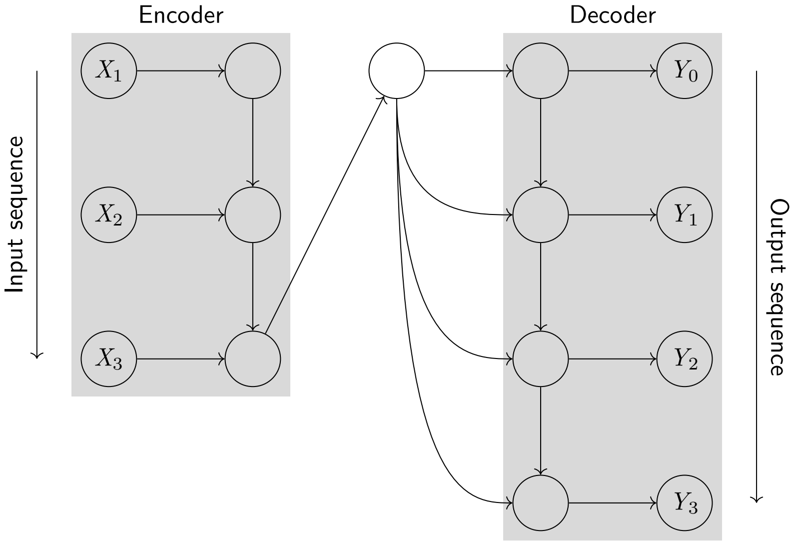
Introduction

- 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

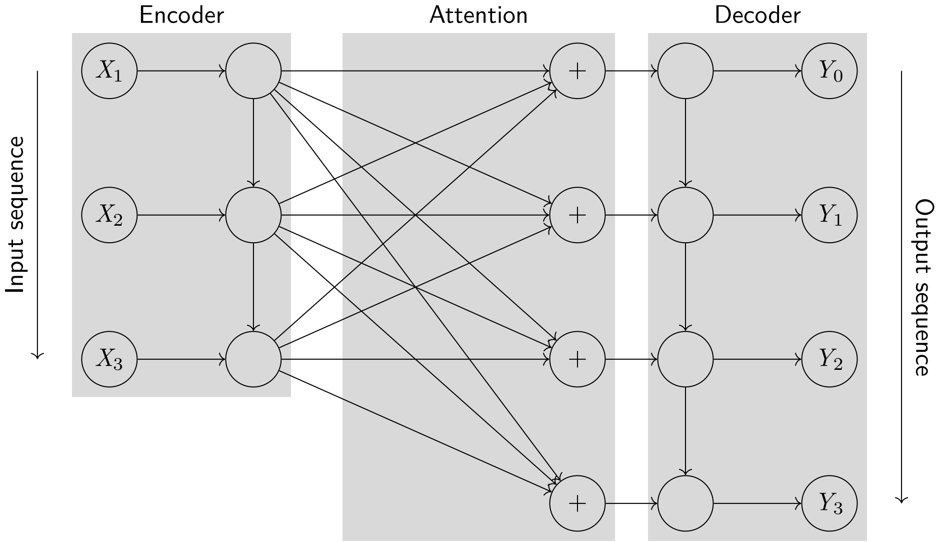
Introduction

- 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

From Encoder-Decoder to Attention



From Encoder-Decoder to Attention



Section 3

Transformer Architecture

Introduction

- BERT is the first succesful model that implements the transformer architecture
- BERT has outperformed the state of the art in many NLP tasks
- Breakthrough in NLP

Introduction

- BERT is the first succesful model that implements the transformer architecture
- BERT has outperformed the state of the art in many NLP tasks
- Breakthrough in NLP
- General idea — Devlin et al. (2019)
  - Encoder-Attention-Decoder architecture (= transformer)
  - Process whole input at once, no sequence labeling! (max. 512 tokens, = bidirectional)
  - Pre-training and fine-tuning on different tasks

Pre-Training and Fine-Tuning

- BERT models are trained on large data sets
- Training one from scratch requires significant resources (time/money)
- Pre-trained models are shared freely
- Recipe: Take a pre-trained model and fine-tune it on your task
  - Pre-trained model contains an abstract language representation

Pre-Training and Fine-Tuning

- BERT models are trained on large data sets
- Training one from scratch requires significant resources (time/money)
- Pre-trained models are shared freely
- Recipe: Take a pre-trained model and fine-tune it on your task
    - Pre-trained model contains an abstract language representation
- Fine-tuning
    - Any language-related task!

BERT Training Tasks

Masked Language Modeling (MLM)

- Sentence-wise
- 15% of the tokens are "masked" by a special token
- Model predicts these, having access to all other tokens

BERT Training Tasks

Masked Language Modeling (MLM)

- Sentence-wise
- 15% of the tokens are “masked” by a special token
- Model predicts these, having access to all other tokens

Next sentence prediction (NSP)

- Two (masked) sentences are concatenated
- Model has to predict wether second sentence follows on the first or not

Section 4

Practical Things and Future Trends

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

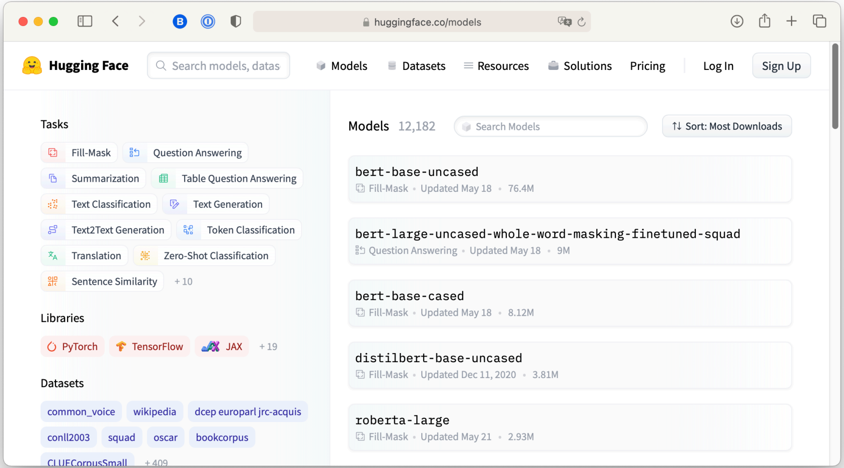
```python
import tensorflow as tf
from transformers import TFAutoModelForSequenceClassification

# 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


huggingface.co/models
Hugging Face
Search models, datas
Models
Datasets
Resources
Solutions
Pricing
Log In
Sign Up
Tasks
Fill-Mask
Question Answering
Summarization
Table Question Answering
Text Classification
Text Generation
Text2Text Generation
Token Classification
Translation
Zero-Shot Classification
Sentence Similarity
+ 10
Libraries
PyTorch
TensorFlow
JAX
+ 19
Datasets
common_voice
wikipedia
dcep europarl jrc-acquis
conll2003
squad
oscar
bookcorpus
CLUECorpusSmall
+ 409
Models 12,182
Search Models
Sort: Most Downloads
bert-base-uncased
Fill-Mask • Updated May 18 • 76.4M
bert-large-uncased-whole-word-masking-finetuned-squad
Question Answering • Updated May 18 • 9M
bert-base-cased
Fill-Mask • Updated May 18 • 8.12M
distilbert-base-uncased
Fill-Mask • Updated Dec 11, 2020 • 3.81M
roberta-large
Fill-Mask • Updated May 21 • 2.93M

Using Large Language Models

- Extracting contextual embeddings
  - `s12-get-bert-features.py`

Using Large Language Models

- Extracting contextual embeddings
  - `s12-get-bert-features.py`
- Predicting the next token / filling in blanks
  - `s12-unmasker.py`

Using Large Language Models

- Extracting contextual embeddings
  - `s12-get-bert-features.py`
- Predicting the next token / filling in blanks
  - `s12-unmasker.py`
- Fine-Tuning to a specific task (using annotated data)
  - `s12-fine-tune-text-classification.py`

Using Large Language Models

- Extracting contextual embeddings
  - `s12-get-bert-features.py`
- Predicting the next token / filling in blanks
  - `s12-unmasker.py`
- Fine-Tuning to a specific task (using annotated data)
  - `s12-fine-tune-text-classification.py`
- Zero-Shot classification (Classify without fine-tuning!)
  - `s12-zero-shot-classification.py`

Using Large Language Models

- Extracting contextual embeddings
  - `s12-get-bert-features.py`
- Predicting the next token / filling in blanks
  - `s12-unmasker.py`
- Fine-Tuning to a specific task (using annotated data)
  - `s12-fine-tune-text-classification.py`
- Zero-Shot classification (Classify without fine-tuning!)
  - `s12-zero-shot-classification.py`
- Few-Shot classification (= "in-context-learning")
  - The new paradigm? Brown et al. (2020)

The Future

- LLMs will not go away, are expensive (money/power/maintenance) and powerful

The Future

- LLMs will not go away, are expensive (money/power/maintenance) and powerful
- Open tasks
  - Proper/rigorous evaluation
    - Humans are good at over-interpreting model output
  - Control: Effectively preventing LLMs from generating bullshit or toxic language
  - Future LLMs: How to gather untainted training data

The Future

- LLMs will not go away, are expensive (money/power/maintenance) and powerful
- Open tasks
  - Proper/rigorous evaluation
    - Humans are good at over-interpreting model output
  - Control: Effectively preventing LLMs from generating bullshit or toxic language
  - Future LLMs: How to gather untainted training data
- How to (properly) use LLMs in scientific areas is still somewhat unclear
  - It's a difference wether a model generates language (with all its ambiguity) or category assignments

Section 5

Summary

Summary

- Motivation: Sequence to sequence tasks (like machine translation)

Encoder-Decoder architecture

- Encoder reads in the input, generates internal representation
- Decoder produces output, consuming internal representation

Attention

- Developed for image classification, then transfered to machine translation
- Let the model learn the relevant input tokens for each output token

Transformer architecture

- Breakthrough in natural language processing
- Pre-training vs. fine-tuning
- Huggingface: Platform to make such models easy to use
  - Good documentation on transformers: huggingface.co/docs/transformers