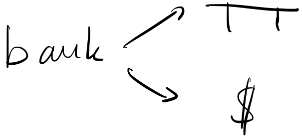
Recap



- Neural networks
  - Layered architecture
  - Output of one layer fed into the next
  - Layer contains neurons, a neuron represents a single calculation
  - Activation functions

Recurrent neural networks for sequence labeling

- Word2Vec training
  - Two architectures
  - Train NN to predict words in contexts
  - Use learned weights as word vectors
  - From Scratch Guide

UNIVERSITÄT
ZU KÖLN

Neural Networks, Part 3: BERT

Sprachverarbeitung (VL + Ü)

Nils Reiter

June 27, 2024

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)
- No recurrent layers – entire input is processed at once with positional embeddings
  - I.e., a fixed number of tokens is fed into the network
- New training paradigm(s) — Devlin et al. (2019)
  - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Section 1

Training Paradigm

Training a Neural Network

- Model training

epochs=30 $\times 2$ => epochs=60

```
model.fit(x_train, y_train, ...)
```

- When is training done?
  - After a number of epochs
  - (or, theoretically, when we reach parameters with minimal loss)
  - I.e.: It's our choice!
  - Nothing prevents us from adding additional epochs after the training

Training a Neural Network

- Model training

```
model.fit(x_train, y_train, ...)
```

- When is training done?
  - After a number of epochs
  - (or, theoretically, when we reach parameters with minimal loss)
  - I.e.: It's our choice!
  - Nothing prevents us from adding additional epochs after the training
- What happens when we supply different data sets?

```
model.fit(x1_train, y1_train, epochs=100)00
model.fit(x2_train, y2_train, epochs=50)
```

Training a Neural Network

- Model training

```
model.fit(x_train, y_train, ...)
```

- When is training done?
  - After a number of epochs
  - (or, theoretically, when we reach parameters with minimal loss)
  - I.e.: It's our choice!
  - Nothing prevents us from adding additional epochs after the training
- What happens when we supply different data sets?

```
model.fit(x1_train, y1_train, epochs=100)
model.fit(x2_train, y2_train, epochs=50)
```

  - Nothing exciting: The model continues to be trained, but with a different dataset🤯

Training a Neural Network

- Model training

```
model.fit(x_train, y_train, ...)
```

- When is training done?
  - After a number of epochs
  - (or, theoretically, when we reach parameters with minimal loss)
  - I.e.: It's our choice!
  - Nothing prevents us from adding additional epochs after the training
- What happens when we supply different data sets?

```
model.fit(x1_train, y1_train, epochs=100)
model.fit(x2_train, y2_train, epochs=50)
```

  - Nothing exciting: The model continues to be trained, but with a different dataset🤯
  - Does the model care if the data sets are about the same task? No. 🤯🤯

Training on Different Datasets

- Not possible with decision trees, naïve Bayes, ...
- Neural networks: No problem
- Why do we want to do that?

Training paradigmas

- Classical: Train data set, evaluate, be happy (or not)
- Neural: Pre-training/Fine-tuning paradigma
  - Pre-train a model on some task
  - Fine-tune it on another

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 2

Encoder-Decoder-Networks

Introduction

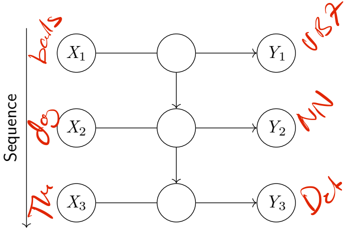
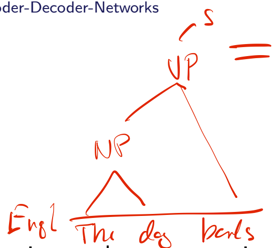


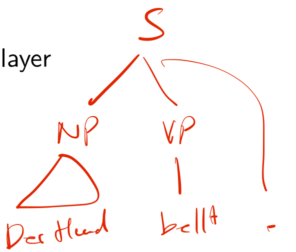
Figure: Neural network with a recurrent layer

- 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

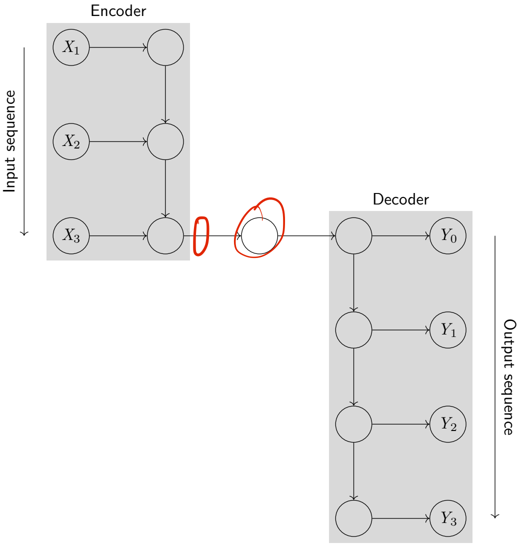


$\Rightarrow \exists x .\ dog(x) \land barks(x)$

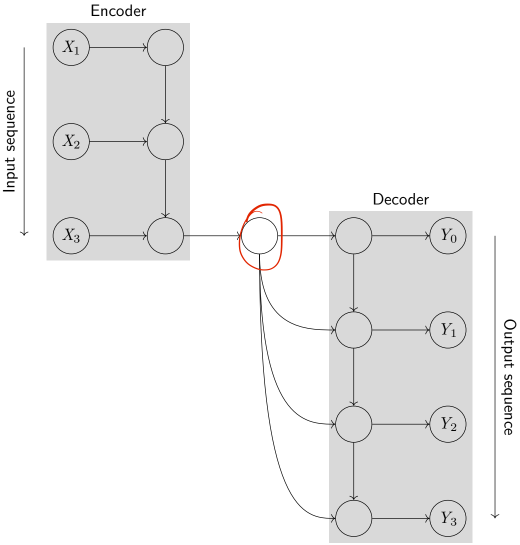


- Network has two parts:
  - Encoder maps from input data to an internal representation
  - Internal representation optionally processed by a regular dense layer
  - Decoder maps from internal representation to the output
- Internal representation
  - Use the output of last recurrent neuron
    - Or 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))
                  simpleRNN

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

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

n = 512

512

0 The => 17

0 dog => 238

0 barks => 795

0 ⋮ ⋮

0

0

0

Section 3

Positional Embeddings

Introduction

- Transformer architecture does not use recurrent connections
- Entire input is consumed at once
  - with dummy tokens, if the sentence is too short
  - BERT context window: 512 tokens
    - I.e.: 512 input neurons, each taking one token index
- But the model still needs to learn something about relative positions

Introduction

- Transformer architecture does not use recurrent connections
- Entire input is consumed at once
  - with dummy tokens, if the sentence is too short
  - BERT context window: 512 tokens
    - I.e.: 512 input neurons, each taking one token index
- But the model still needs to learn something about relative positions

Example

- Input 1: »Cologne is also part of the Rhine-Ruhr metropolitan region, the second biggest metropolitan region by GDP in the European Union.«
- Input 2: »The second biggest metropolitan region by GDP in the European Union is the Rhine-Ruhr region.«
- Model should learn that »biggest« and »region« are related, even though they are in different positions

➔ Positional Embeddings

BERT Input

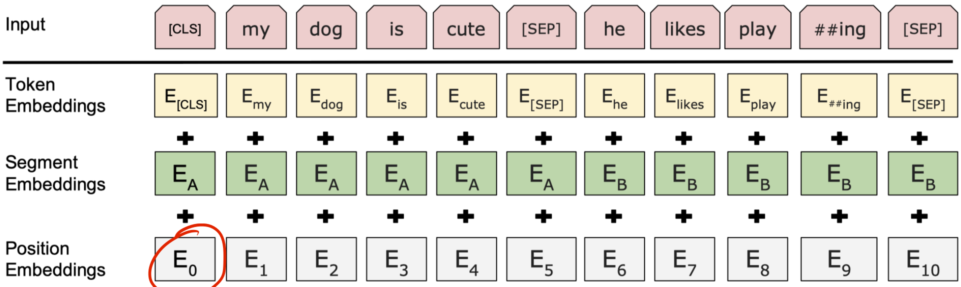


Figure: BERT input representation (Devlin et al., 2019)

Position Embeddings

- Each position is encoded as a vector
  - I.e., position 1 has a vector that is different from position 2, etc.
- Position vectors have the same length as the token vectors (allowing summation)
  - After token embeddings and position embeddings have been added, they represent a token at it's position
- BERT: Position embeddings are learned, just like other embeddings

Section 4

Attention

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

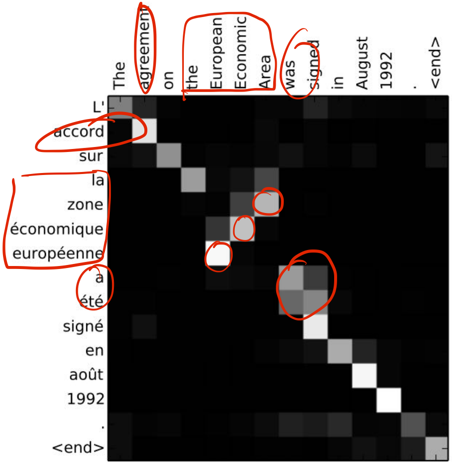


Figure: Attention paid by a neural machine translation network (Bahdanau et al., 2015)

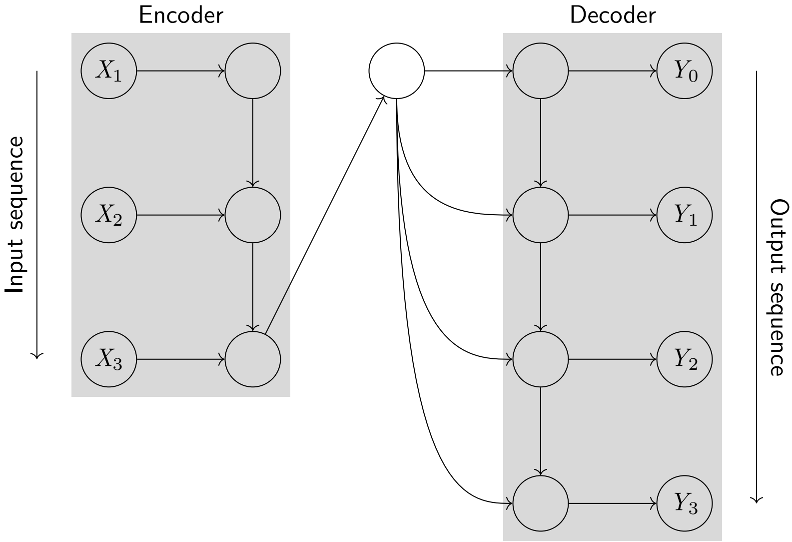
Introduction

- 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

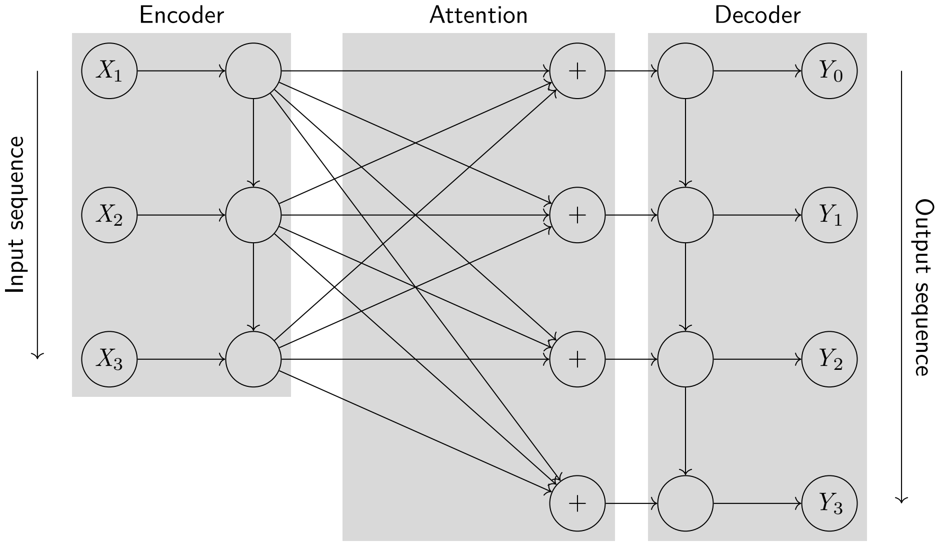
Introduction

- 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« → »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 5

Practical Things and Future Trends

Hugging Face

Introduction

- An AI company that provides
  - A Python library for transformers 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

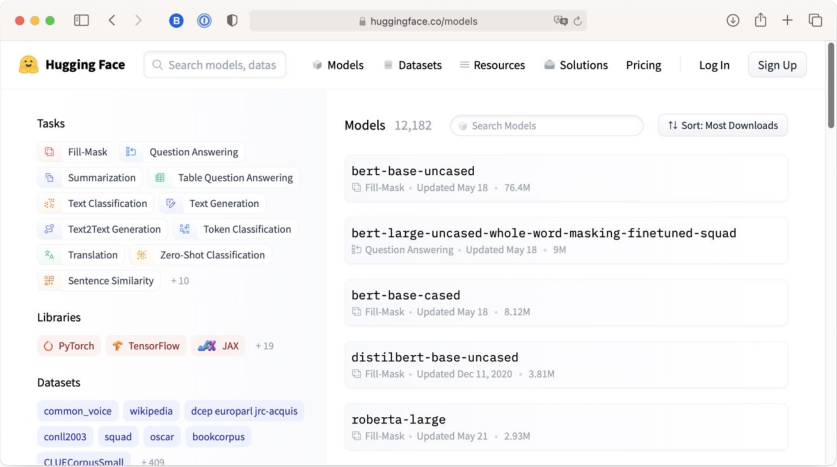
```
import tensorflow as tf
from transformers import TFAutoModelForSequenceClassification

# 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)

References I

- Bahdanau, Dzmitry/Kyunghyun Cho/Yoshua Bengio (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/Yann LeCun. URL: http://arxiv.org/abs/1409.0473.
- Brown, Tom B./Benjamin Mann/Nick Ryder/Melanie Subbiah/Jared Kaplan/Prafulla Dhariwal/Arvind Neelakantan/Pranav Shyam/Girish Sastry/Amanda Askell/Sandhini Agarwal/Ariel Herbert-Voss/Gretchen Krueger/Tom Henighan/Rewon Child/Aditya Ramesh/Daniel M. Ziegler/Jeffrey Wu/Clemens Winter/Christopher Hesse/Mark Chen/Eric Sigler/Mateusz Litwin/Scott Gray/Benjamin Chess/Jack Clark/Christopher Berner/Sam McCandlish/Alec Radford/Ilya Sutskever/Dario Amodei (2020). *Language Models are Few-Shot Learners*. DOI: 10.48550/arXiv.2005.14165. arXiv: 2005.14165 [cs.CL].

References II

Devlin, Jacob/Ming-Wei Chang/Kenton Lee/Kristina Toutanova (2019). »BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding«. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

Sutskever, Ilya/Oriol Vinyals/Quoc V Le (2014). »Sequence to Sequence Learning with Neural Networks«. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani/M. Welling/C. Cortes/N. Lawrence/K.Q. Weinberger. Vol. 27. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Vaswani, Ashish/Noam Shazeer/Niki Parmar/Jakob Uszkoreit/Llion Jones/Aidan N. Gomez/Lukasz Kaiser/Illia Polosukhin (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].