

Deep Learning for Text 2

Applied Text Mining

Ayoub Bagheri

Recap: RNN in Python

```
1 embedding_dim = 100
2 model = Sequential()
3 model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
4 model.add(layers.LSTM(100, dropout=0.2, recurrent_dropout=0.2))
5 model.add(layers.Dense(10, activation='relu'))
6 model.add(layers.Dense(5, activation='softmax'))
7 model.compile(optimizer='adam',
8                 loss='categorical_crossentropy',
9                 metrics=['accuracy']))
10 model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
embedding (Embedding)	(None, 100, 100)	3811100
lstm (LSTM)	(None, 100)	80400
dense (Dense)	(None, 10)	1010
dense_1 (Dense)	(None, 5)	55
=====		
Total params: 3,892,565		
Trainable params: 3,892,565		
Non-trainable params: 0		

Lecture plan

1. Convolutional Neural Networks
2. Transformers
3. BERT

Convolutional Neural Network (CNN)

- ▶ Intuition: Neural network with specialized connectivity structure
 - ▶ Stacking multiple layers of feature extractors, low-level layers extract local features, and high-level layers extract learn global patterns.
- ▶ There are a few distinct types of layers:
 - ▶ **Convolution Layer**: detecting local features through filters (discrete convolution)
 - ▶ **Pooling Layer**: merging similar features

Convolution layer

- ▶ The core layer of CNNs
- ▶ Convolutional layer consists of a set of filters
- ▶ Each filter covers a spatially small portion of the input data
- ▶ Each filter is convolved across the dimensions of the input data, producing a multidimensional **feature map**.
- ▶ As we convolve the filter, we are computing the dot product between the parameters of the filter and the input.
- ▶ **Deep Learning algorithm:** During training, the network corrects errors and filters are **learned**, e.g., in Keras, by adjusting weights based on **Stochastic Gradient Descent**, **SGD**.
- ▶ The key architectural characteristics of the convolutional layer is **local connectivity** and **shared weights**.

Convolution without padding

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5x5 input.

1	0	1
0	1	0
1	0	1

3x3 filter/kernel/feature detector. 3x3 convolved feature/
activation map/feature map

4	3	4
2	4	3
2	3	4

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5x5 input.

1	0	1
0	1	0
1	0	1

3x3 filter/kernel/feature detector. 3x3 convolved feature/
activation map/feature map

4	3	4
2	4	3
2	3	4

Convolution with padding

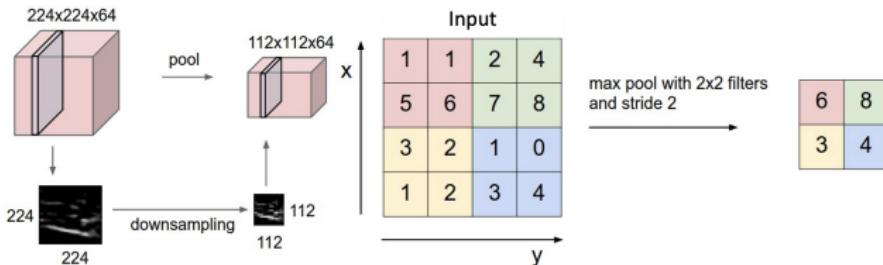
4x4 input. 3x3 filter. Stride = 1. 2x2 output.

5x5 input. 3x3 filter. Stride = 1. 5x5 output.

https://github.com/vdumoulin/conv_arithmetic

Pooling layer

- ▶ Intuition: to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting
- ▶ Pooling partitions the input image (or documents) into a set of non-overlapping rectangles (n -grams) and, for each such sub-region, outputs the maximum value of the features in that region.



Pooling (down sampling)

2	2	4	4
2	4	8	4
4	4	4	1
6	10	3	4

Max pooling

4	8
10	4

Mean Pooling

2.5	5
6	3

- The new size after pooling!

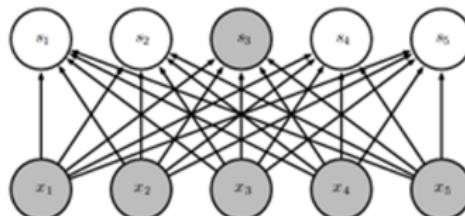
Convolutional neural network

For processing data with a **grid-like** or array topology:

- ▶ 1-D convolution: text data, sequence data, time-series data, sensor signal data
- ▶ 2-D convolution: image data
- ▶ 3-D convolution: video data

Other layers

- ▶ The convolution, and pooling layers are typically used as a set. Multiple sets of the above layers can appear in a CNN design.
- ▶ After a few sets, the output is typically sent to one or two **fully connected layers**.
 - ▶ A fully connected layer is an ordinary neural network layer as in other neural networks.
 - ▶ Typical activation function is the sigmoid function.
 - ▶ Output is typically class (classification) or real number (regression).



Other layers

- ▶ The final layer of a CNN is determined by the research task.
- ▶ Classification: Softmax Layer

$$P(y = j | \mathbf{x}) = \frac{e^{w_j \cdot \mathbf{x}}}{\sum_{k=1}^K e^{w_k \cdot \mathbf{x}}}$$

- ▶ The outputs are the probabilities of belonging to each class.
- ▶ Regression: Linear Layer

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$$

- ▶ The output is a real number.

What hyperparameters do we have in a CNN model?

CNN for Text

CNN

Main CNN idea for text:

Compute vectors for n-grams and group them afterwards

Example: “Utrecht summer school is in Utrecht” compute vectors for:

Utrecht summer, summer school, school is, is in, in Utrecht,

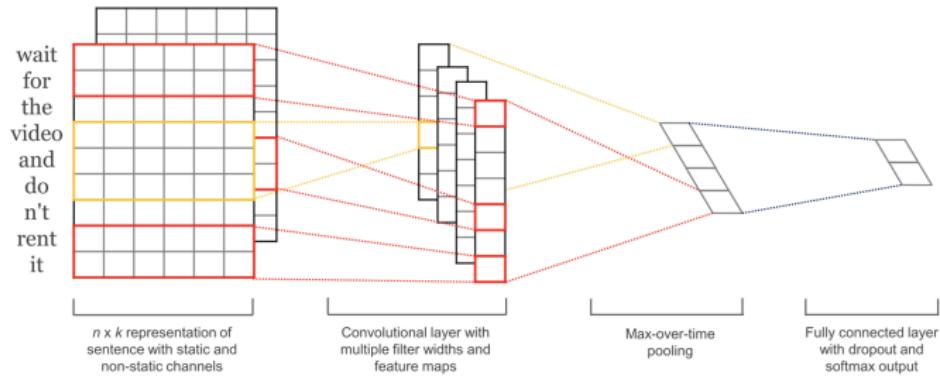
Utrecht summer school, summer school is, school is in, is in

Utrecht, Utrecht summer school is, summer school is in, school is

in Utrecht, Utrecht summer school is in, summer school is in

Utrecht, Utrecht summer school is in Utrecht

CNNs for sentence classification



Kim, Y. "Convolutional Neural Networks for Sentence Classification", EMNLP (2014)

sliding over 3, 4 or 5 words at a time

<https://arxiv.org/pdf/1408.5882.pdf>

Data sets (1)

- ▶ **MR**: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews (Pang and Lee, 2005). url:
<https://www.cs.cornell.edu/people/pabo/movie-review-data/>
- ▶ **SST-1**: Stanford Sentiment Treebank—an extension of MR but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative), re-labeled by Socher et al. (2013). url:
<https://nlp.stanford.edu/sentiment/>
- ▶ **SST-2**: Same as SST-1 but with neutral reviews removed and binary labels.
- ▶ **Subj**: Subjectivity dataset where the task is to classify a sentence as being subjective or objective (Pang and Lee, 2004).

Data sets (2)

- ▶ **TREC**: TREC question dataset—task involves classifying a question into 6 question types (whether the question is about person, location, numeric information, etc.) (Li and Roth, 2002). url: <https://cogcomp.seas.upenn.edu/Data/QA/QC/>
- ▶ **CR**: Customer reviews of various products (cameras, MP3s etc.). Task is to predict positive/negative reviews (Hu and Liu, 2004). url:
<https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>
- ▶ **MPQA**: Opinion polarity detection subtask of the MPQA dataset (Wiebe et al., 2005). url:
https://mpqa.cs.pitt.edu/corpora/mpqa_corpus/

Datasets' statistics

Data	c	l	N	V	V _{pre}	Test
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

Table 1: Summary statistics for the datasets after tokenization. c : Number of target classes. l : Average sentence length. N : Dataset size. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of pre-trained word vectors. $Test$: Test set size (CV means there was no standard train/test split and thus 10-fold CV was used).

CNN variations

- **CNN-rand:** Our baseline model where all words are randomly initialized and then modified during training.
- **CNN-static:** A model with pre-trained vectors from word2vec. All words—including the unknown ones that are randomly initialized—are kept static and only the other parameters of the model are learned.
- **CNN-non-static:** Same as above but the pre-trained vectors are fine-tuned for each task.
- **CNN-multichannel:** A model with two sets of word vectors.

Similar words

		Most Similar Words for	
		Static Channel	Non-static Channel
<i>bad</i>	<i>good</i>		<i>terrible</i>
	<i>terrible</i>		<i>horrible</i>
	<i>horrible</i>		<i>lousy</i>
	<i>lousy</i>		<i>stupid</i>
<i>good</i>	<i>great</i>		<i>nice</i>
	<i>bad</i>		<i>decent</i>
	<i>terrific</i>		<i>solid</i>
	<i>decent</i>		<i>terrific</i>
<i>n't</i>	<i>os</i>		<i>not</i>
	<i>ca</i>		<i>never</i>
	<i>ireland</i>		<i>nothing</i>
	<i>wo</i>		<i>neither</i>
<i>!</i>	2,500		2,500
	<i>entire</i>		<i>lush</i>
	<i>jez</i>		<i>beautiful</i>
	<i>changer</i>		<i>terrific</i>
,	<i>decasia</i>		<i>but</i>
	<i>abysmally</i>		<i>dragon</i>
	<i>demise</i>		<i>a</i>
	<i>valiant</i>		<i>and</i>

Results

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	89.6
CNN-non-static	81.5	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	—	—	—	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	—	—	—	—
RNTN (Socher et al., 2013)	—	45.7	85.4	—	—	—	—
DCNN (Kalchbrenner et al., 2014)	—	48.5	86.8	—	93.0	—	—
Paragraph-Vec (Le and Mikolov, 2014)	—	48.7	87.8	—	—	—	—
CCAE (Hermann and Blunsom, 2013)	77.8	—	—	—	—	—	87.2
Sent-PARSER (Dong et al., 2014)	79.5	—	—	—	—	—	86.3
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	—	—	93.6	—	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	—	—	93.4	—	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	—	—	93.6	—	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	—	—	—	—	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	—	—	—	—	—	82.7	—
SVM _S (Silva et al., 2011)	—	—	—	—	95.0	—	—

CNN with Keras in Python

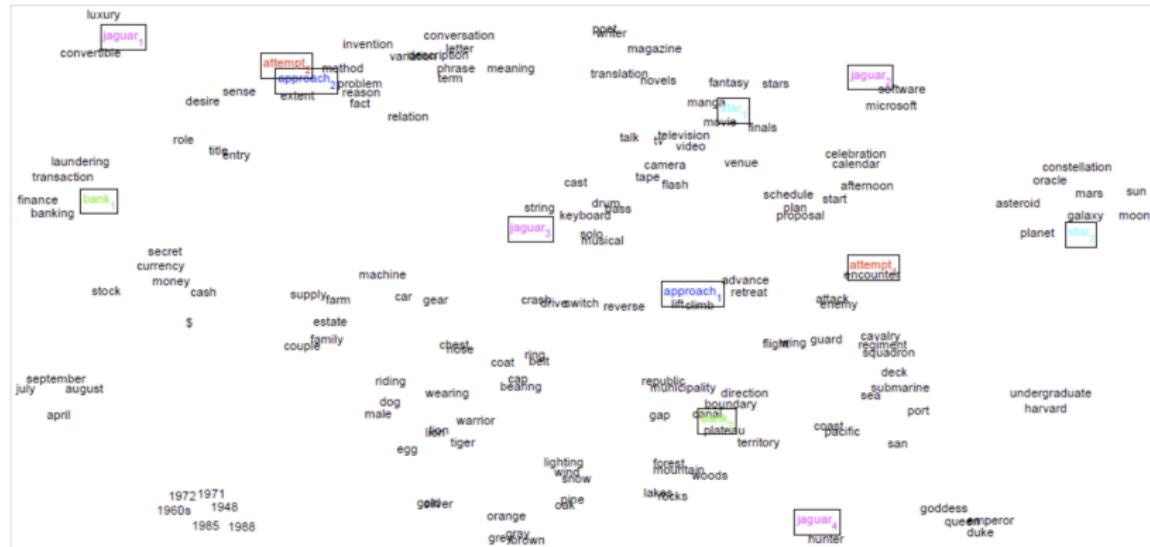
```
model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(5, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
<hr/>		
embedding_2 (Embedding)	(None, 100, 100)	3811100
conv1d (Conv1D)	(None, 96, 128)	64128
global_max_pooling1d (Global)	(None, 128)	0
dense_4 (Dense)	(None, 10)	1290
dense_5 (Dense)	(None, 5)	55
<hr/>		
Total params: 3,876,573		
Trainable params: 3,876,573		
Non-trainable params: 0		

Contextual Word Embeddings & Transformers

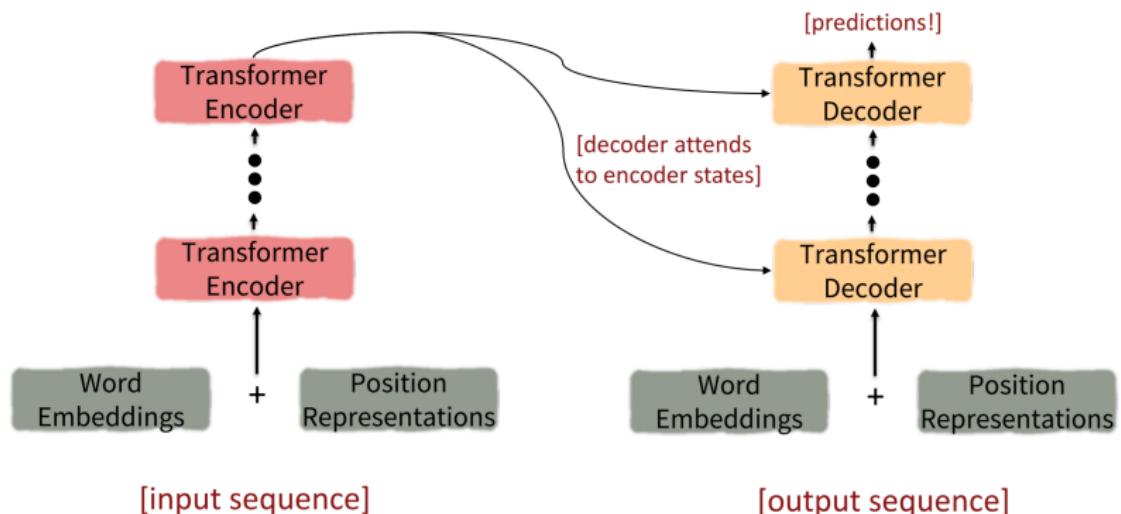
Contextual Word Embeddings



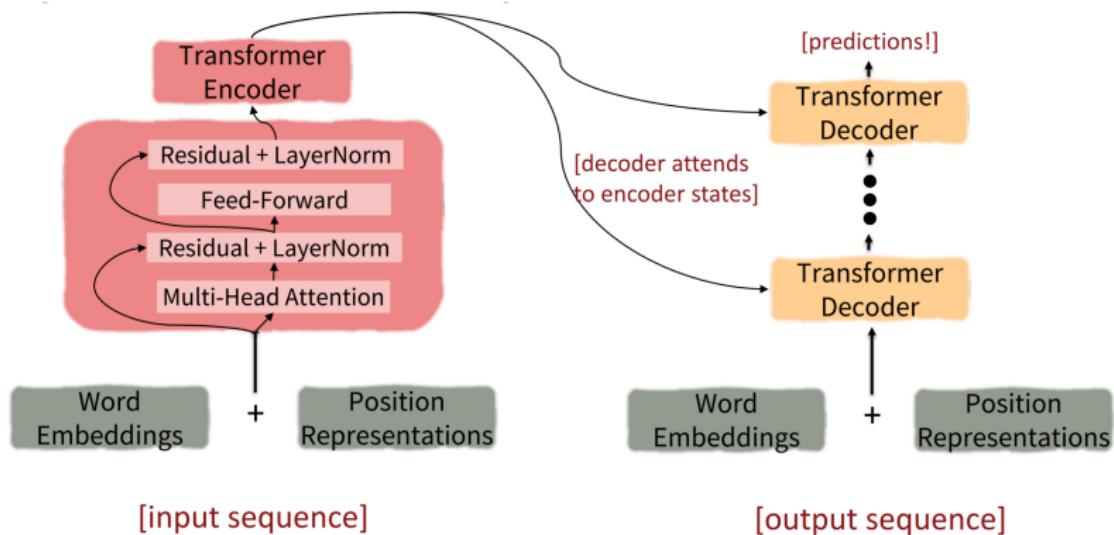
Transformers

- ▶ A transformer adopts an encoder-decoder architecture.
- ▶ Transformers were developed to solve the problem of sequence transduction, or neural machine translation. That means any task that transforms an input sequence to an output sequence.
- ▶ More details on the architecture and implementation:
 - ▶ <https://arxiv.org/abs/1810.04805>
 - ▶ <http://nlp.seas.harvard.edu/2018/04/03/attention.html>
 - ▶ <https://jalammar.github.io/illustrated-transformer/>

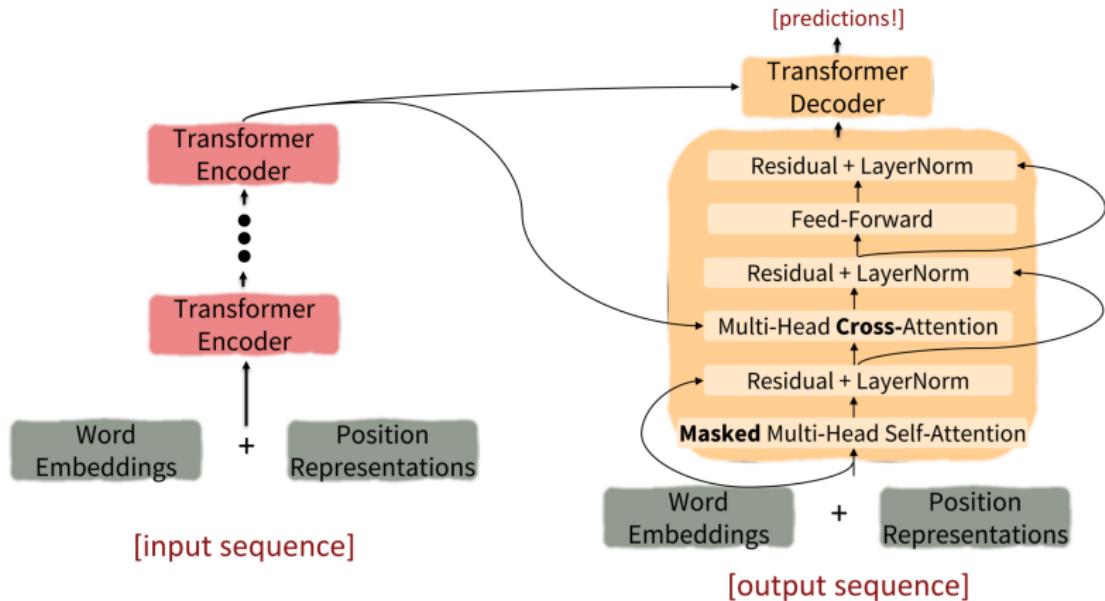
The Transformer Encoder-Decoder



The Transformer Encoder-Decoder



The Transformer Encoder-Decoder



Transformers

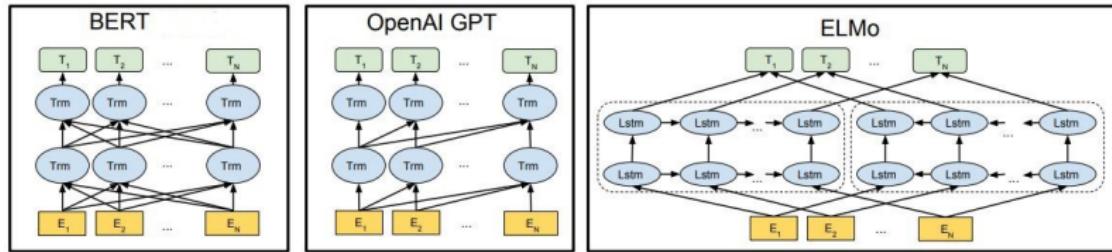


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

BERT: Bidirectional Encoder Representations from Tranformers

BERT: Bidirectional Encoder Representations from Tranformers

What kinds of things does pretraining learn?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language:

Talk to Transformer: <https://app.inferkit.com/demo>

- ▶ Utrecht University is located in . . .

Transformers

Transformers

Transformers

What kinds of things does pretraining learn?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language:

- ▶ **Basic arithmetic:** I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ...

Transformers

What kinds of things does pretraining learn?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language:

- ▶ **Reasoning:** Garry went into the kitchen to make some tea. Standing next to Garry, Carrie pondered her destiny. Carrie left the ...

Transformers

Transformers

Transformers

Transformers

- ▶ Write with Transformer: <https://transformer.huggingface.co/>
- ▶ Talk to Transformer: <https://app.inferkit.com/demo>
- ▶ Transformer model for language understanding:
<https://www.tensorflow.org/text/tutorials/transformer>
- ▶ Pretrained models:
https://huggingface.co/transformers/pretrained_models.html

Summary

Summary

- ▶ Convolutional Neural Networks
- ▶ Transformers
 - ▶ “Small” models like BERT have become general tools in a wide range of settings
 - ▶ GPT-3 has 175 billion parameters
- ▶ These models are still not well-understood

Practical 7