Chapter 10: BERT (& the Sesame Street)

Matthias Aßenmacher

January 20, 2021

Key facts of BERT ▸ Devlin et al. (2018)

B*idirectional* **E***ncoder* **R***epresentations from* **T***ransformers:*

- Bidirectionally contextual model
- Introduces new self-supervised objective(s)
- Completely replaces recurrent architectures by Self-Attention
  + simultaneously able to include bidirectionality

Predecessors of BERT

2013 - word2vec
Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2013

Predecessors of BERT

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework.

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2013 **01/2018**

January 2018 - ULMFiT

The first transfer learning architecture (**U**niversal **L**anguage **M**odel **Fi**ne-**T**uning) was proposed by **Howard and Ruder (2018)**.
An embedding layer at the bottom of the network was complemented by three AWD-LSTM layers (Merity et al., 2017) and a softmax layer for pre-training.

A **Unidirectional contextual** model since no biLSTMs are used.

Predecessors of BERT

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

February 2018 - ELMo

Guys from **AllenNLP** developed a bidirectionally contextual framework by proposing ELMo (**E**mbeddings from **L**anguage **Mo**dels; **Peters et al., 2018**).

Embeddings from this architecture are the (weighted) combination of the intermediate-layer representations produced by the biLSTM layers.

January 2018 - ULMFiT

The first transfer learning architecture (**U**niversal **L**anguage **M**odel **Fi**ne-**T**uning) was proposed by **Howard and Ruder (2018)**.
An embedding layer at the bottom of the network was complemented by three AWD-LSTM layers (Merity et al., 2017) and a softmax layer for pre-training.

A **Unidirectional contextual** model since no biLSTMs are used.

Predecessors of BERT

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

February 2018 - ELMo

Guys from **AllenNLP** developed a bidirectionally contextual framework by proposing ELMo (**E**mbeddings from **L**anguage **Mo**dels; **Peters et al., 2018**).

Embeddings from this architecture are the (weighted) combination of the intermediate-layer representations produced by the biLSTM layers.

January 2018 - ULMFiT

The first transfer learning architecture (**U**niversal **L**anguage **M**odel **Fi**ne-**T**uning) was proposed by **Howard and Ruder (2018)**.
An embedding layer at the bottom of the network was complemented by three AWD-LSTM layers (Merity et al., 2017) and a softmax layer for pre-training.

A **Unidirectional contextual** model since no biLSTMs are used.

June 2018 - OpenAI GPT

Radford et al., 2018 abandon the use of LSTMs. The combine multiple Transformer decoder block with a standard language modelling objective for pre-training.

Compared to ELMo it is just **unidirectionally contextual**, since it uses only the decoder side of the Transformer. On the other hand it is **end-to-end trainable** (cf. ULMFiT) and embeddings do not have to be extracted like in the case of ELMo.

Predecessors of BERT

2013 | 01/2018 | 02/2018 | 06/2018 | 10/2018

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

January 2018 - ULMFiT

The first transfer learning architecture (**U**niversal **L**anguage **M**odel **Fi**ne-**T**uning) was proposed by **Howard and Ruder (2018)**.
An embedding layer at the bottom of the network was complemented by three AWD-LSTM layers (Merity et al., 2017) and a softmax layer for pre-training.

A **Unidirectional contextual** model since no biLSTMs are used.

February 2018 - ELMo

Guys from **AllenNLP** developed a bidirectionally contextual framework by proposing ELMo (**E**mbeddings from **L**anguage **Mo**dels; **Peters et al., 2018**).

Embeddings from this architecture are the (weighted) combination of the intermediate-layer representations produced by the biLSTM layers.

June 2018 - OpenAI GPT

Radford et al., 2018 abandon the use of LSTMs. The combine multiple Transformer decoder block with a standard language modelling objective for pre-training.

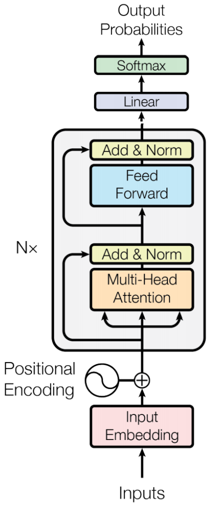
Compared to ELMo it is just **unidirectionally contextual**, since it uses only the decoder side of the Transformer. On the other hand it is **end-to-end trainable** (cf. ULMFiT) and embeddings do not have to be extracted like in the case of ELMo.

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

Core of BERT ▸ Devlin et al. (2018)



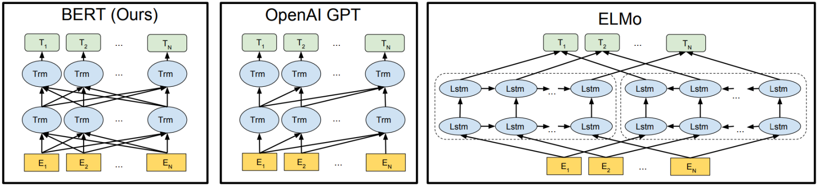
Source: ▸ Vaswani et al. (2017)

A remark on Self-Supervision

Causality is an issue!

- *Remember:* Input and target sequences are the same
  $\rightarrow$ We modify the input to create a meaningful task
- A sequence is used to predict itself again
- Bidirectionality at a lower layer would allow a word to see itself at later hidden layers
  $\rightarrow$ The model would be allowed to cheat!
  $\rightarrow$ This would not lead to meaningful internal representations

GPT vs. ELMo vs. BERT



Source: Devlin et al. (2018)

Major architectural differences:

- ELMo uses two separate unidirectional models to achieve bidirectionality
  $\rightarrow$ Only "*shallow*" bidirectionality
- GPT is not bidirectional, thus no issues concerning causality
- BERT combines the best of both worlds:

$$\textit{Self-Attention} + (\textit{Deep})\ \textit{Bidirectionality}$$

Masked Language Modeling (MLM)

First of all:

- It has nothing to do with Masked Self-Attenion
  $\rightarrow$ Masked Self-Attention is an architectural detail in the decoder of a Transformer, i.e. used by e.g. GPT
- Masked Self-Attention as a way to induce causality in the decoder
- MLM is a modeling objective introduced to couple Self-Attention and (deep) bidirectionality without violating causality

Masked Language Modeling (MLM) ctd.

Masked Language Modeling:

- **Training objective:**

  Given a sentence, predict [MASK]ed tokens

- **Generation of samples:**

  Randomly replace* a fraction of the words by [MASK]

  *Sample 15% of the tokens; replace 80% of them by [MASK], 10% by a random token & leave 10% unchanged

- **Input:**

The	quick	brown	[MASK]	jumps	over	the	[MASK]	dog	.

- **Targets:**

  (*fox*, *lazy*)

Masked Language Modeling (MLM) ctd.

Discrepancy between pre-training & fine-tuning:

- `[MASK]`-token as central part of pre-training procedure
- `[MASK]`-token does not occur during fine-tuning
- **Modified pre-training task:**
  Predict 15% of the tokens of which only 80% have been replaced by `[MASK]`
  - 80% of the selected tokens:
    `The quick brown fox` $\rightarrow$ `The quick brown [MASK]`
  - 10% of the selected tokens:
    `The quick brown fox` $\rightarrow$ `The quick brown went`
  - 10% of the selected tokens:
    `The quick brown fox` $\rightarrow$ `The quick brown fox`

Next Sentence Prediction (NSP)

Next Sentence Prediction:

- **Training objective:**

  Given two sentences, predict whether s_2 follows s_1

- **Generation of samples:**

  Randomly sample* negative examples (cf. word2vec)

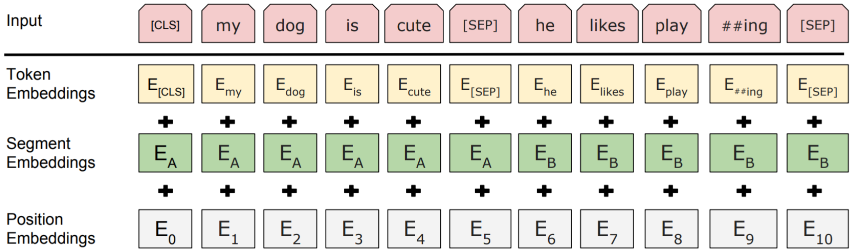
  *50% of the time the second sentence is the actual next sentence, 50% of the time it is a randomly sampled sentence

- **Full Input:**

[CLS]	The	[MASK]	is	quick	.	[SEP]	
It	jumps	over	the	[MASK]	dog	.	[SEP]

- [CLS] token as sequence representation for classification
- [SEP] token for separation of the two input sequences

BERT's input embeddings



Source: Devlin et al. (2018)

- Byte-Pair encoding Sennrich et al. (2016) for the inputs
  $\rightarrow$ Vocabulary of 30.000 tokens

Pre-Training BERT

Ingredients:

- Massive lexical resources (BooksCorpus + Eng. Wikipedia) $\rightarrow$ 13 GB in total
- Train for approximately* 40 epochs
- 4 (16) Cloud TPUs for 4 days for the BASE (LARGE) variant
- 12 (24) Transformer encoder blocks with an embedding size of $E = 768$ (1024) and a hidden layer size $H = E$, $H/64 = 12$ (16) attention heads are used and the feed-forward size is set to $4H$ $\rightarrow$ 110M (340M) model parameters in total for $BERT_{Base}$ ($BERT_{Large}$)
- Loss function:

$$Loss_{BERT} = Loss_{MLM} + Loss_{NSP}$$

*1.000.000 steps on batches of 256 sequences with a sequence length of 512 tokens

Pre-Training BERT – Maximum sequence length

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

not cool cool

Source: Vaswani et al. (2017)

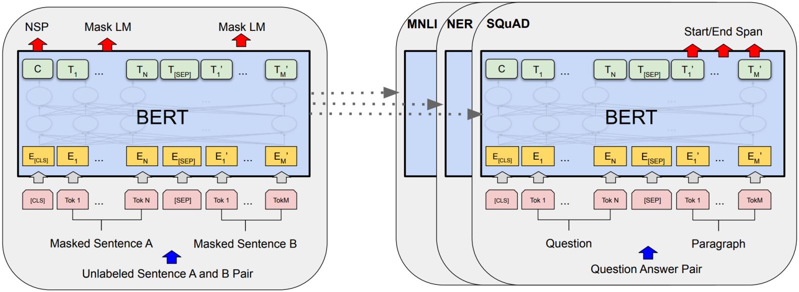
Limitation:

- BERT can only consume sequences of up to 512 tokens
- Two sentences for NSP are sampled such that

$$length_{sentenceA} + length_{sentenceB} \leq 512$$

- Reason: Computational complexity of Transformer scales quadratically with the sequence length
  $\rightarrow$ Longer sequences are disproportionally expensive

Fine-Tuning BERT



Pre-training Fine-Tuning

Source: Devlin et al. (2018)

Fine-Tuning BERT

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Source: Devlin et al. (2018)

- Performance of BERT on the GLUE Benchmark
- Beats all of the previous state-of-the-art models
- In the meantime: Other models better than BERT

Successors of BERT

October 2018 - BERT
BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

10/2018

Successors of BERT

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

10/2018

02/2019

February 2019 - GPT2

Radford et al., 2019 massively scale up their GPT model from 2018 (up to 1.5 billion parameters). Despite the size, there are only smaller architectural changer, thus it remains a **unidirectionally contextual** model.

Controversial debate about this model, since OpenAI (at first) refuses to make their pre-trained architecture publicly available due to concerns about "malicious applications".

Successors of BERT

10/2018 | 02/2019 | 06/2019

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

February 2019 - GPT2

Radford et al., 2019 massively scale up their GPT model from 2018 (up to 1.5 billion parameters). Despite the size, there are only smaller architectural changer, thus it remains a **unidirectionally contextual** model.

Controversial debate about this model, since OpenAI (at first) refuses to make their pre-trained architecture publicly available due to concerns about “malicious applications”.

June 2019 - XLNet

Yang et al., 2019 design a new pre-training objective in order to overcome some weaknesses they spotted in the one used by BERT.

The use **Permutation Language Modelling** to avoid the discrepancy between pre-training and fine-tuning introduced by the artificial MASK token.

Successors of BERT

10/2018 | 02/2019 | 06/2019 | 07/2019

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

February 2019 - GPT2

Radford et al., 2019 massively scale up their GPT model from 2018 (up to 1.5 billion parameters). Despite the size, there are only smaller architectural changer, thus it remains a **unidirectionally contextual** model.

Controversial debate about this model, since OpenAI (at first) refuses to make their pre-trained architecture publicly available due to concerns about "malicious applications".

June 2019 - XLNet

Yang et al., 2019 design a new pre-training objective in order to overcome some weaknesses they spotted in the one used by BERT.

The use **Permutation Language Modelling** to avoid the discrepancy between pre-training and fine-tuning introduced by the artificial MASK token.

July 2019 - RoBERTa

Liu et al., 2019 concentrate on improving the original BERT architecture by (1) careful hyperaparameter tuning (2) abandoning the additional Next Sentence Prediction objective (3) increasing the pre-training corpus *massively*.

Other approaches now more and more concentrate on improving, down-scaling or understanding BERT. A new research direction called **BERTology** emerges.

Successors of BERT

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

February 2019 - GPT2

Radford et al., 2019 massively scale up their GPT model from 2018 (up to 1.5 billion parameters). Despite the size, there are only smaller architectural changer, thus it remains a **unidirectionally contextual** model.

Controversial debate about this model, since OpenAI (at first) refuses to make their pre-trained architecture publicly available due to concerns about "malicious applications".

June 2019 - XLNet

Yang et al., 2019 design a new pre-training objective in order to overcome some weaknesses they spotted in the one used by BERT.

The use **Permutation Language Modelling** to avoid the discrepancy between pre-training and fine-tuning introduced by the artificial MASK token.

July 2019 - RoBERTa

Liu et al., 2019 concentrate on improving the original BERT architecture by (1) careful hyperaparameter tuning (2) abandoning the additional Next Sentence Prediction objective (3) increasing the pre-training corpus *massively*.

Other approaches now more and more concentrate on improving, down-scaling or understanding BERT. A new research direction called **BERTology** emerges.

October 2019 - T5

T5 (**Raffel et al., 2019**) a complete **encoder-decoder** Transformer based architecture (**text-to-text transfer transformer**).

They approach transfer learning by transforming all inputs as well as all outputs to strings and fine-tuned their model simultaneously on data sets with multiple different tasks.

BERTology ▸ Rodgers et al., 2020

Post-BERT architectures:

- Most architectures still rely on either an encoder- *or* a decoder-style type of model (e.g. ▸ GPT2, ▸ XLNet)
- *BERTology:* Many papers/models which aim at ..
  - .. explaning BERT (e.g. ▸ Coenen et al., 2019, ▸ Michel et al., 2019)
  - .. improving BERT (▸ RoBERTa, ▸ ALBERT)
  - .. making BERT more efficient (▸ ALBERT, ▸ DistilBERT)
  - .. modifying BERT (▸ BART)
- Overview on many different papers: https://github.com/tomohideshibata/BERT-related-papers

RoBERTa ▸ Liu et al., 2019

Improvements in Pre-Training:

- Authors claim that BERT is seriously undertrained
- Change of the `MASK`ing strategy
  $\rightarrow$ BERT masks the sequences once before pre-training
  $\rightarrow$ RoBERTa uses dynamic `MASK`ing
  $\Rightarrow$ RoBERTa sees the same sequence `MASK`ed differently
- RoBERTa does not use the additional NSP objective during pre-training
- 160 GB of pre-training resources instead of 13 GB
- Pre-training is performed with larger batch sizes (8k)

Dynamic vs. Static Masking ▸ Liu et al., 2019

Static Masking (BERT):

- Apply `MASK`ing procedure to pre-training corpus once
- (additional for BERT: Modify the corpus for NSP)
- Train for approximately 40 epochs

Dynamic Masking (RoBERTa):

- Duplicate the training corpus *ten* times
- Apply `MASK`ing procedure to each duplicate of the pre-training corpus
- Train for 40 epochs
- Model sees each training instance in ten different "versions" (each version four times) during pre-training

RoBERTa ‣ Liu et al., 2019

Architectural differences:

- Architecture (layers, heads, embedding size) identical to BERT
- 50k token BPE vocabulary instead of 30k
- Model size differs (due to the larger embedding matrix)
  $\Rightarrow \sim$ 125M (360M) for the BASE (LARGE) variant

Performance differences:

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

Source: Liu et al. (2019)

Note: Liu et al. (2019) report the accuracy for QQP while Devlin et al. (2018) report the F1 score (cf. results displayed on slide 15); XLNet: see next Chapter.

ALBERT ▸ Lan et al., 2019

Changes in the architecture:

- Disentanglement of embedding size E and hidden layer size H
  $\rightarrow$ WordPiece-Embeddings (size E) context-independent
  $\rightarrow$ Hidden-Layer-Embeddings (size H) context-dependent
  $\Rightarrow$ Setting $H >> E$ enlargens model capacity without increasing the size of the embedding matrix,
  since $O(V \times H) > O(V \times E + E \times H)$ if $H >> E$.
- Cross-Layer parameter sharing
- Change of the pre-training NSP loss
  $\rightarrow$ Introduction of *Sentence-Order Prediction* (SOP)
  $\rightarrow$ Positive examples created alike to those from NSP
  $\rightarrow$ Negative examples: Just swap the ordering of sentences
- $n - gram$ masking for the MLM task

ALBERT [Lan et al., 2019]

Performance differences:

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

Source: Lan et al. (2019)

Notes:

- In General: Smaller model size (because of parameter sharing)
- Nevertheless: Scale model up to almost similar size (`xxlarge` version)
- Strong performance compared to BERT

Using BERT & Co.

Native implementations:

- BERT: *https://github.com/google-research/bert*
- RoBERTa: *https://github.com/pytorch/fairseq/tree/master/examples/roberta*
- ALBERT: *https://github.com/google-research/ALBERT*

Drawbacks:

- Different frameworks use for the implementations
- Different programming styles
  $\rightarrow$ Adaption of different models to custom problems can sometimes lead to a lot of redundant work

Example: Fine-tune native BERT on MRPC

Command line:

```
!python run_classifier.py \
--task_name=MRPC \
--do_train=true \
--do_eval=true \
--data_dir=$GLUE_DIR/MRPC \
--vocab_file=$BERT_BASE_DIR/vocab.txt \
--bert_config_file=$BERT_BASE_DIR/bert_config.json \
--init_checkpoint=$BERT_BASE_DIR/bert_model.ckpt \
--max_seq_length=128 \
--train_batch_size=32 \
--learning_rate=2e-5 \
--num_train_epochs=3.0 \
--output_dir=/tmp/mrpc_output/
```

Pre-trained architectures @ `transformers`

Unified API for state-of-the-art architectures:

- 32+ pre-trained architectures (as of January 22, 2021)
- Models in 100+ languages available (as of January 22, 2021)
- Implementations in PyTorch as well as TensorFlow 2.0
- Unified naming model parts and fine-tuning procedures
- Docs: *https://huggingface.co/transformers/index.html*

Which different building blocks available?

- Model architecture
- Custom tokenizers for each architecture
- (Sets of) Pre-trained weights for an architecture
- Pre-defined heads for fine-tuning on common tasks

Pre-trained architectures @ transformers

Pre-trained weights:

```
<model name>-<version>-<cased/uncased>
```

Load tokenizer & tokenize a sentence:

```python
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-cased")
sentence = "Hello guys, welcome to the course."

ids = tokenizer.encode(sentence, add_special_tokens = True)
ids

# [101, 8667, 3713, 117, 7236, 1106, 1103, 1736, 119, 102]

[tokenizer.convert_ids_to_tokens(id) for id in ids]

# ['[CLS]', 'Hello', 'guys', ',', 'welcome', 'to', 'the',
#  'course', '.', '[SEP]']
```

Pre-trained architectures @ `transformers`

Tokenization all in one:

- BERT requires inputs of fixed length → Padding
- `[PAD]` tokens should not receive Attention weights

```
tokenizer.encode_plus(sentence,
                      add_special_tokens = True,
                      max_length = 12,
                      pad_to_max_length = True,
                      return_attention_mask = True,
                      return_tensors = 'pt'
                   )

# {'input_ids': tensor([[ 101, 8667, 3713,  117, 7236,
#                      1106, 1103, 1736,  119,  102,    0,
     0]]),
# 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0]]),
# 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   0, 0]])}
```

Pre-trained architectures @ transformers

Load a model architecture:

```
from transformers import BertForSequenceClassification
mod = BertForSequenceClassification.from_pretrained("bert-
    base-cased",

    num_labels = 3)
```

Inspect the dimensionality of the model:

```
params = list(mod.parameters())

## size of the embedding layer
params[0].shape

# torch.Size([28996, 768])

## size of the classification layer
params[200].shape

# torch.Size([3])
```

Pre-trained architectures @ `transformers`

Pipelines:

- Extremely high-level API included in `transformers`
- Available for a couple of different downstream tasks
- Ingredients of a pipeline:
  - *Encoding:* Tokenization of the inputs
  - Inferency by a chosen model
  - *Decoding:* Use model output to generate target values

```
from transformers import pipeline

pipeline(<task name>, model = <model name>,
         tokenizer = <tokenizer name>)
```

- `model` and `tokenizer` are optional arguments
  $\rightarrow$ If not provided, some internal defaults are used

Pre-trained architectures @ `transformers`

Available tasks (as of January 22, 2021):

- Feature Extraction (`feature-extraction`)
- Sentiment Analysis (`"sentiment-analysis"`)
- Named entity recognition (`"ner"`)
- Question Answering (`"question-answering"`)
- [MASK]-filling (`"fill-mask"`)
- Summarization (`"summarization"`)
- Translation (`"translation-xx-to-yy"`)
- seq2seq Text Generation (`"text2text-generation"`)
- Text Generation (`"text-generation"`)
- Zero-Shot Classification (`"zero-shot-classification"`)
- Multi-turn Conversation (`"conversation"`)

Pre-trained architectures @ `transformers`

Exemplary task (with default model):

```
from transformers import pipeline

pipe_classif = pipeline("sentiment-analysis")
pipe_classif(sentence)

# [{'label': 'POSITIVE', 'score': 0.99960136}]

pipe_classif("I absolutely hate this!")

# [{'label': 'NEGATIVE', 'score': 0.9992645}]
```

Default:

- DistilBERT model
- `base, uncased`
- Fine-tuned on SST-2 data set

Pre-trained architectures @ transformers

Use other models than the default:

```
pipe_fill = pipeline("fill-mask",
    model = "bert-large-cased",
    tokenizer = BertTokenizer.from_pretrained("bert-large-
   cased"))
pipe_fill("I like " + pipe_fill.tokenizer.mask_token + "
   football.")

# [{'sequence': '[CLS] I like playing football. [SEP]',
#  'score': 0.464905500411987З,
#  'token': 1773},
# {'sequence': '[CLS] I like watching football. [SEP]',
#  'score': 0.19629190862178802,
#  'token': 2903},
# {'sequence': '[CLS] I like the football. [SEP]',
#  'score': 0.10121186822652817,
#  'token': 1103},
# {'sequence': '[CLS] I like American football. [SEP]',
#  'score': 0.053604859858751З,
#  'token': 1237}]
```

Other languages than English

Multilingual models:

- BERT also available as multilingual model
- Top 100 languages with the largest Wikipedias
- Re-weighting of training data (favor low-resoure languages)
- 110k shared WordPiece vocabulary
- Released in a `base, cased` version
- *https://github.com/google-research/bert/blob/master/multilingual.md*

Monolingual models:

- Specifically trained for each language separately
- Examples for German:
  - *deepset.ai*
  - *Bayerische Staatsbibliothek*

Other languages than English

```
pipe_fill_ger = pipeline("fill-mask",
    model="bert-base-german-cased",
    tokenizer=AutoTokenizer.from_pretrained("bert-base-
    german-cased"))
pipe_fill_ger("Der Himmel ist " + pipe_fill.tokenizer.
    mask_token)

# [{'sequence': '[CLS] Der Himmel ist blau [SEP]',
#   'score': 0.18068572878837585,
#   'token': 8516},
# {'sequence': '[CLS] Der Himmel ist leer [SEP]',
#   'score': 0.10186842083930969,
#   'token': 12101},
# {'sequence': '[CLS] Der Himmel ist frei [SEP]',
#   'score': 0.04153556749224663,
#   'token': 1409}]
```

Further reading

- See reference to GitHub-repo on Slide 17