BERT

The Architecture

Learning goals

- Understand the use of the transformer encoder in this model
- Understand the architectural components

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 – 01/2018 – 02/2018

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.

PREDECESSORS OF BERT

2013 → 01/2018 → 02/2018 → 06/2018

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 **Mo**del **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.

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.

CONTEXT: ULMFIT AND GPT

Shortcomings of ELMo:

- No adaption of the Embeddings to target domain/task
- Sequential nature of LSTMs: Not fully parallelizable

Alleviations/Alternatives:

- ULMFiT ▸ Howard and Ruder, 2018 is a uni-directional LSTM which is fine-tuned as a whole model on data from the target domain/task.
- GPT ▸ Radford et al., 2018 is a Transformer decoder which is fine-tuned as a whole model on data from the target domain/task.

All three still not sufficient:

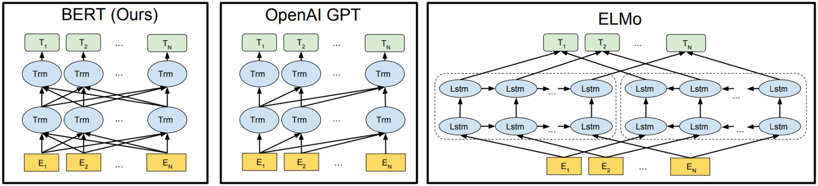
- *Bidirectionally contextual:* Only ELMo
- *Parallelizable:* Only GPT
- *Fine-tune whole model:* Only ULMFiT & GPT

BERT: KEY FACTS

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

- Bidirectionally contextual model
  $\rightarrow$ The embeddings of a single token depend on its left- and on its right-side context (similar to ELMo, but better)
- Completely replaces recurrent architectures by Self-Attention
  $\rightarrow$ parallelizable
- Model can fine-tuned as a whole

ELMO VS. GPT VS. BERT



Source: Devlin et al., 2019

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:

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

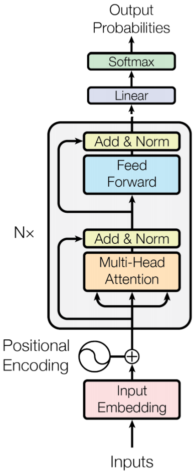
BERT: KEY FACTS

- New self-supervised objective(s)

  $\rightarrow$ MLM as *necessity* for the architecture to work

  $\rightarrow$ Next-Sentence-Prediction as complementary objective (cf. next section)
- Transformer *encoder* as backbone of the architecture
- 110M (340M) parameters in total for $BERT_{Base}$ ($BERT_{Large}$)
  - 12 (24) Transformer encoder blocks
  - Embedding size of $E = 768$ (1024)
  - Hidden layer size $H = E$
  - $A = H/64 = 12$ (16) attention heads
  - Feed-forward size is set to $4H$

CORE OF BERT – TRANSFORMER ENCODER



▸ Vaswani et al. (2017)

A REMARK ON "CAUSALITY"

Causality is an issue!

- Goal: Learn contextual representations for words/tokens
- *Self-Supervision:* Input and target sequence are the same
  $\rightarrow$ We modify the input to create a meaningful task
- ⚠ Unconstrained Self-Attention makes using the LM objective infeasible
- **Question:** Why is this the case?

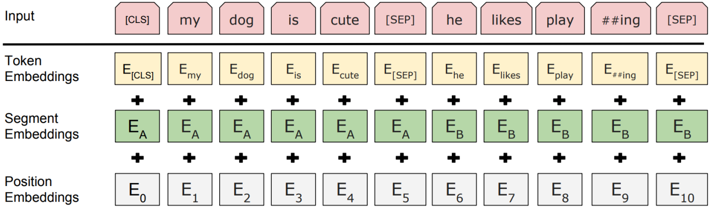
A REMARK ON "CAUSALITY"

Causality is an issue!

- Goal: Learn contextual representations for words/tokens
- *Self-Supervision:* Input and target sequence are the same
  $\rightarrow$ We modify the input to create a meaningful task
- ⚠ Unconstrained Self-Attention makes using the LM objective infeasible
- **Question:** Why is this the case?

  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

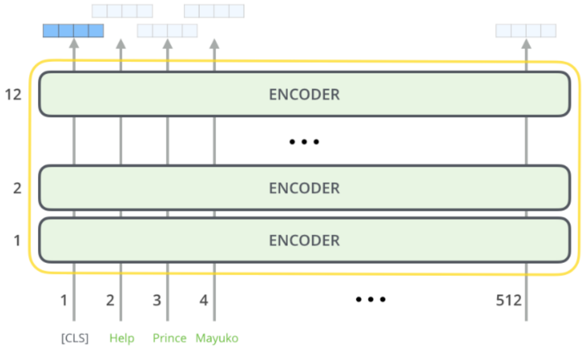
BERT – INPUT EMBEDDINGS



▸ Source: Devlin et al., 2019

- Two concatenated sentences as input
- WordPiece tokenization ▸ Wu et al., 2016 for the inputs
  $\rightarrow$ Vocabulary of 30.000 tokens
- *Learned* segment + position embeddings
- Special `[CLS]` and `[SEP]` tokens

BERT – ALL EMBEDDINGS



Source: Jay Alammar

- One embedding per token per layer
- Non-contextual embeddings in the very first embedding layer
- More contextualization deeper into the model

BERT – THE ROLE OF [CLS] AND [SEP]

Why deliberately include extra “words“?

- The `[CLS]` token serves as an overall embedding for representing the whole sequence
  $\rightarrow$ Later on (cf. next chapter) BERT can thus used for classifying whole sequences
  $\rightarrow$ Can be extracted and used for clustering or similar
- The `[SEP]` (short for “‘separator‘) token serves as a “signal“ for the model when used for taks on pairs of sequences

Note:

- One further “special“ token: `[UNK]` for representing unknown tokens or symbols (so the don’t “break“ the model)

TWO DIFFERENT BERT VERSIONS

There are two different BERT versions, namely BERT base and large. Depending on the architecture we get different parameter counts

▸ Source: Devlin et al., 2019

- **BERT base:**
  - $n_{layers} = 12$, we have 12 encoder layers
  - $n_{heads} = 12$, this will not affect the parameter count since they perfectly split d_{model} across the heads
  - $d_{model} = 768$, thats the embedding dimension

- **BERT large:**
  - $n_{layers} = 24$
  - $n_{heads} = 16$
  - $d_{model} = 1024$

ENCODER PARAMETER COUNT

From the chapter about the transformer parameter count we already know the number of parameters for one Encoder layer: $12 \cdot d_{model}^2$

$$N_{Encoder} = n_{layers} \cdot 12 \cdot d_{model}^2$$

- **BERT base**:

$$N_{Encoder} = 12 \cdot 12 \cdot 768^2 = 84,934,656$$

- **BERT large**:

$$N_{Encoder} = 24 \cdot 12 \cdot 1024^2 = 301,989,888$$

By increasing d_{model} *and* n_{layers} *we more than tripled the number of Encoder parameters!*

EMBEDDING PARAMETER COUNT

Similar to the Transformer chapter we also have to consider the paramters from the embeddings:

- Let V be the vocabulary size, M the maximum sequence length and S the number of segments
- BERT has three kinds of embeddings, which are all learned:
  - $V \times d_{model}$ token embeddings
  - $S \times d_{model}$ segment embeddings
  - $M \times d_{model}$ position embeddings
- From the BERT paper we know $V = 30000$, $S = 2$ and $M = 512$
- **BERT base**:

  $$N_{Embedding} = 30000 \times 768 + 2 \times 768 + 512 \times 768 = 23,434,752$$

- **BERT large**:

  $$N_{Embedding} = 30000 \times 1024 + 2 \times 1024 + 512 \times 1024 = 31,246,336$$

FINAL PARAMETER COUNT

Now we just have to sum up both parts to get the final parameter count:

- **BERT base**:

$$N_{Total} = 84,934,656 + 23,434,752 = 108,369,408 \approx 110M$$

- **BERT large**:

$$N_{Total} = 301,989,888 + 31,246,336 = 333,236,224 \approx 340M$$