Hands-on Introduction to Deep Learning with PyTorch

Natural Language Processing with Transformers

Rafael Sarmiento and Rocco Meli
ETHZürich / CSCS

Lugano, February 28th - March 1st 2024

ETH zürich
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Outline

- Language modeling
- Tokenization
- The Transformer Model
- [lab] Fine-tuning BERT for Q&A

Language Modeling

"The cat sat on the <...>"

↓

model

↓

{ "mat": 0.3, "rug": 0.2,

"chair": 0.2, ... }

- A **language model** is a model that learns the structure and patterns of a language from a corpus of text data
- Its primary function is to predict the probability of a **sequence** of words or characters occurring in a given context

Language Modeling

"The cat <MASK> on the mat"

↓

model

↓

{ "sat": 0.3, "slept": 0.2,
"rested": 0.2, ... }

- A **language model** is a model that learns the structure and patterns of a language from a corpus of text data
- Its primary function is to predict the probability of a **sequence** of words or characters occurring in a given context
- Can be found in **next token prediction** or **masked language modeling** tasks where the objective is to predict a missing **token** based on the context from both its left and right sides

Language Model Pretraining and Fine-Tuning

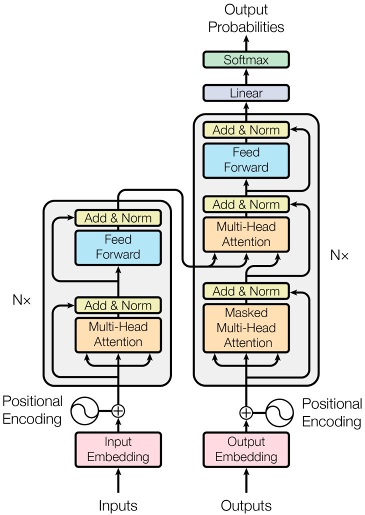
These days LMs start their life by being **pretrained** on objectives such as next token prediction (NTP) or masked language modeling (MLM) and then they are **fine-tuned** for different downstream tasks such as machine translation, text classification, Q&A and more

Pretraining

- large datasets
- specific pretraining task
- the model learns general features and representations of the language

Fine-tuning

- the pretrained LM is further trained where it's parameters are adjusted to better suit the target task
- smaller datasets
- different task relevant to the target application



- State of the art LMs are based on the **transformer** architecture introduced in the paper ”Attention is All You Need” by Vaswani et al. in 2017
- The core component of transformers is the **self-attention** mechanism
  - Allows the model to weigh the importance of different tokens in a sequence
  - Enables the model to capture long-range dependencies and contextual information efficiently
  - Efficient parallelization
- Transformers have been successfully applied to a wide range of NLP tasks such as text classification, machine translation, named entity recognition, sentiment analysis, Q&A and text generation

A. Vaswani *et al.* Attention Is All You Need

- **HuggingFace** is a platform for Natural Language Processing development
- Home to the `transformers`, `tokenizers` and `datasets` packages
- Easy-to-use interfaces to access an extensive library of pre-trained models
- Open-source community-driven development

Tokenization

Tokenization is the process of converting text into smaller components, known as tokens. These tokens can be words, characters, or subwords.

"The cat sat on the mat"

↓

"[CLS]"	"the"	"cat"	"sat"	"on"	"the"	"mat"	"[SEP]"
101	1996	4937	2938	2006	1996	13523	102

White Space Tokenization

- Splits text based on spaces.

$$\text{"The cat sat on the mat"} \rightarrow \left\{ \begin{array}{l} \text{"The"} \\ \text{"cat"} \\ \text{"sat"} \\ \text{"on"} \\ \text{"the"} \\ \text{"mat"} \end{array} \right\}$$

- Not always effective, especially when dealing with multi-word phrases, contractions, hyphenated words, etc

Subword Tokenization

- Techniques like Byte Pair Encoding (BPE), Byte-level BPE and WordPiece
- BPE merges frequent pairs iteratively until the desired vocabulary size is reached. For instance, the input sequence "unexpectedness" is tokenized as ["un" "expected" "ness"] with the BPE tokenizer used for GPT-2
- Helps in dealing with rare words and morphologically rich languages, handling out-of-vocabulary words, better generalization to unseen words, efficient encoding of morphological variations

Why Tokenization?

- Reduction of vocabulary size
- Improves handling of rare words
- Adaptable to different languages and domains

Word embeddings

"The cat sat on the mat"

[101, 1996, 4937, 2938, 2006, 1996, 13523, 102]

[CLS] [SEP]

$$\left[\, \vec{e}_1,\ \vec{e}_2, \dots \boxed{\vec{e}_{1996}}, \dots \boxed{\vec{e}_{2006}}, \dots\ \vec{e}_{5000}, \dots\ \vec{e}_{\text{vocab\_size}} \right]$$

1996 2006

$$\left[\, \vec{e}_{101},\ \vec{e}_{1996},\ \vec{e}_{4937},\ \vec{e}_{2938},\ \vec{e}_{2006},\ \vec{e}_{1996},\ \vec{e}_{13523},\ \vec{e}_{102} \,\right]$$

Word embeddings

- Enable algorithms to represent words as continuous vectors
- Word embeddings are parameters of the transformer model
- During training the embeddings vectors are adjusted to better capture the relation between tokens
- Word embeddings are based on the distributional hypothesis: words with similar meanings tend to occur in similar contexts

Contextual representations

”time flies like an arrow”

”fruit flies like banana”

Self-attention (basic)

$$\text{seq} = \left[\; \vec{e}_1 \quad \vec{e}_2 \quad \vec{e}_3 \quad \ldots \quad \vec{e}_n \;\right]$$

$$\vec{e}_i^{\,*} = \sum_{k}^{\text{seq}} a_{ij}\vec{e}_j$$

$$\text{seq}^* = \left[\; \vec{e}_1^{\,*} \quad \vec{e}_2^{\,*} \quad \vec{e}_3^{\,*} \quad \ldots \quad \vec{e}_n^{\,*} \;\right]$$

- Allows the model to weigh the importance of different tokens in a sequence
- Enables the model to capture long-range dependencies and contextual information efficiently
- Efficient parallelization
- The **attention weights** a_{ij} are defined as a similarity function for a pair of embeddings $\vec{e}_i$ and $\vec{e}_j$ such as $\vec{e}_i \cdot \vec{e}_j$ or $\text{softmax}_j(\vec{e}_i \cdot \vec{e}_j)$

Scaled Dot-Product Attention

$$a_{ij} = \text{softmax}_j(\vec{e}_i \cdot \vec{e}_j) \quad \rightarrow \quad a_{ij} = \text{softmax}_j\left(\frac{W_\mathsf{q}\vec{e}_i \cdot W_\mathsf{k}\vec{e}_j}{\sqrt{d_k}}\right)$$

$$\vec{e}_i^{\,*} = \sum_k^{\text{seq}} a_{ij}\vec{e}_j \quad \rightarrow \quad \vec{e}_i^{\,*} = \sum_k^{\text{seq}} a_{ij} W_\mathsf{v}\vec{e}_j$$

- $W_\mathsf{k}\vec{e}_i$, $W_\mathsf{q}\vec{e}_i$ and $W_\mathsf{v}\vec{e}_i$ are called keys, queries and values respectively
- They are just projections of the embeddings into spaces of lower dimensions and as a result the $\vec{e}_i^{\,*}$ are vectors of a smaller size than $\vec{e}_i$
- The matrices W_k, W_q and W_v are parameters of the model

Multi-head Attention

$$\vec{e}_i^* = \sum_k^{\text{seq}} a_{ij} W_{\text{v}} \vec{e}_j$$

$$\vec{e}_i^* = \sum_k^{\text{seq}} a_{ij} W_{\text{v}} \vec{e}_j$$

$$\ldots$$

$$\vec{e}_i^* = \sum_k^{\text{seq}} a_{ij} W_{\text{v}} \vec{e}_j$$

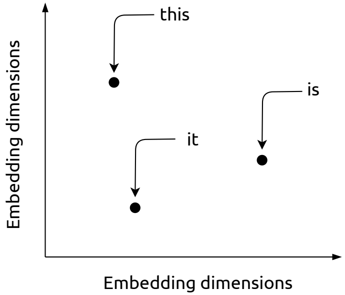
$$\begin{bmatrix} \vec{e}_1^* & \vec{e}_2^* & \vec{e}_3^* & \cdots & \vec{e}_{\text{seq\_len}}^* \\ \vec{e}_1^* & \vec{e}_2^* & \vec{e}_3^* & \cdots & \vec{e}_{\text{seq\_len}}^* \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vec{e}_1^* & \vec{e}_2^* & \vec{e}_3^* & \cdots & \vec{e}_{\text{seq\_len}}^* \end{bmatrix} \updownarrow \text{concat}$$

- The attention computation is done multiple independent times in different **attention heads** for the same input sequence using different W_{k}, W_{q} and W_{v} matrices
- The idea behind this is that each head can capture diverse patterns and dependencies in the input sequence (attending to different features of the sequence)
- The different final $\vec{e}_j^*$ are concatenated (typically) recovering the original embedding dimension

Position-wise Feed Forward Layer

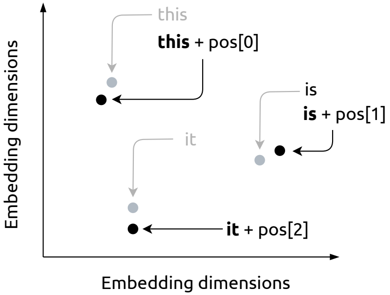
- Consists of two linear transformations separated by a non-linear activation function, typically GELU
- The first linear transformation projects the input from the embedding dimension to a higher-dimensional space (typically four times the size of the embeddings) and the second one projects the intermediate representation back to the original embedding dimension
- It's applied independently to each position in the sequence
- Most of the memorization is thought to happen in these layers and their sizes vary from model to model

Positional encoding



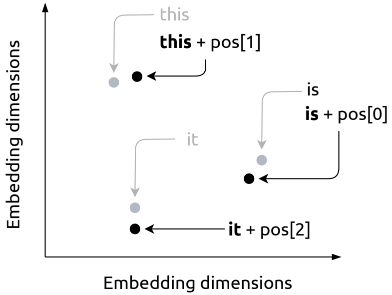
- Consider the sequence "this is it" tokenized as ["this", "is", "it"]

Positional encoding



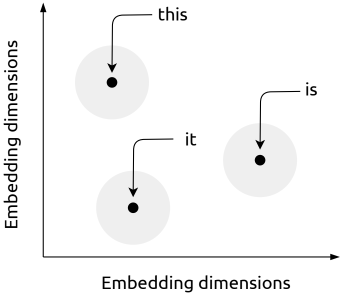
- Consider the sequence "this is it" tokenized as ["this", "is", "it"]
- Adding the positional encodings, shifts the word embedding of each token adding the information of where in the sentence each token is

Positional encoding



- Consider the sequence "this is it" tokenized as ["this", "is", "it"]
- Adding the positional encodings, shifts the word embedding of each token adding the information of where in the sentence each token is
- In the sequence "is this it", each final embedding is different compared to "this is it"

Positional encoding



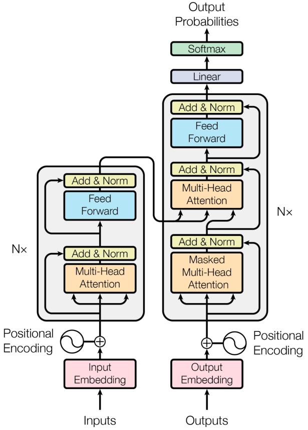
- Consider the sequence "this is it" tokenized as ["this", "is", "it"]
- Adding the positional encodings, shifts the word embedding of each token adding the information of where in the sentence each token is
- In the sequence "is this it", each final embedding is different compared to "this is it"
- By seeing many examples during training, the model learns to identify the tokens in different positions
- With learned positional embeddings, during training, the model adjusts them to better capture the relationships between tokens in different positions

Masked self-attention

$$\begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

- Masks allow the model to selectively attend to certain tokens while disregarding others based on a predefined mask
- Masked attention selectively prevents tokens from attending to future tokens by creating contextual embeddings of a token by using only tokens at its left side
- Necessary for next token prediction (autoregresive modeling)

Transformer architectures

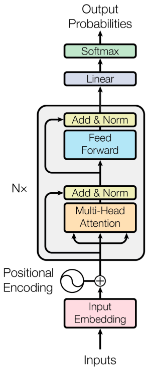


Encoder-decoder

- Suitable for **sequence to sequence** tasks such as machine translation and text summarization
- Combines an encoder that focus on the input sequence and a decoder that helps in generating the output sequence
- Contains self-attention, masked self-attention and cross-attention blocks
- BART and T5 are examples of models of this category

A. Vaswani *et al.* Attention Is All You Need

Transformer architectures

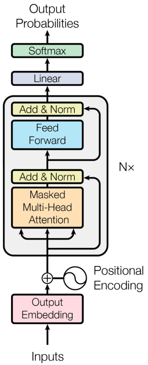


Encoder-only

- Primarily used to obtain numerical representation from input sequences of text such as text classification and text extraction for Q&A task
- Contains self-attention blocks and the contextual representations for a given token has information from both the left and right parts of the sequence(bidirectional attention)
- BERT and its variants, like RoBERTa and DistilBERT belong to this category
- This is often called bidirectional attention

A. Vaswani *et al.* Attention Is All You Need

Transformer architectures



Decoder-only

- Primarily used for autoregressive generation tasks where the output sequence is generated token by token (iterative prediction of the most probable next word)
- Contains masked self-attention blocks and as a result the contextual representations for a given token depends only on the part of the sequence at its left side (causal attention)
- GPT models belong to this category

A. Vaswani *et al.* Attention Is All You Need

[lab] Finetuning MobileBERT for Q&A

We are going to fine-tune MobileBERT implemented by HuggingFace for the Q&A by text-extraction task with the The Stanford Question Answering Dataset (SQuAD).

Thank you for your attention!