Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica

Lesson 13

Encoder-only Transformers

Nicola Capuano and Antonio Greco

DIEM – University of Salerno

Outline

- Encoder-only transformer
- BERT
- Practice on token classification

and named entity recognition

Encoder-only transformer

Encoder-only transformer

- You need the whole transformer architecture if the task you have to solve requires the transformation of a sequence into a sequence of different length
    - Example: translation between different languages (the original application of the Transformer model)
- For a task where you want to **transform a sequence into another sequence of the same length**, you can use just the Encoder part of the transformer
    - In this case, the vectors $z_1,\ldots, z_t$ will be your output vectors, and you can compute the loss function directly on them
- For a task where you want to **transform a sequence into a single value** (e.g., a sequence classification task), you can use just the Encoder part of the transformer
    - You add a special START value as the first element x_1 of the input
    - You take the corresponding output value z_1 as the result of your transformation, and compute your loss function only on it

BERT

BERT

- BERT is a language model introduced by Google Research in 2018 for language understanding. It is the acronym of Bidirectional Encoder Representation from Transformers
- It uses only the Encoder part from the Transformer model. BERT-base has 12 stacked encoder blocks (110M parameters), while BERT-large 24 (340M parameters).
- BERT is pre-trained to use "bidirectional" context (i.e. successive words, as well as previous ones, for understanding a word in a sentence)
- It is designed to be used as a pre-trained base for Natural Language Processing tasks (after fine tuning)

BERT input encoding

- BERT uses a **WordPiece** tokenizer as its tokenization method.

- **Subword-Based**: WordPiece tokenizes text at the subword level rather than using full words or individual characters. This helps BERT handle both common words and rarer or misspelled words by breaking them into manageable parts.
- **Vocabulary Building**: The WordPiece tokenizer builds a vocabulary of common words and subword units (e.g., "playing" could be tokenized as "play" and "##ing").
- **Unknown Words**: Rare or out-of-vocabulary words can be broken down into familiar subwords. For example, "unhappiness" could be split into "un," "happy," and "##ness."

BERT input encoding

- **Splitting**: Sentences are split into tokens based on whitespace, punctuation, and common prefixes (like "##").
- BERT requires certain special tokens:
  - **[CLS]**: A classification token added at the beginning of each input sequence.
  - **[SEP]**: A separator token used to mark the end of a sentence (or between sentences in the case of sentence-pair tasks).
- For example, the sentence "I'm feeling fantastic!" might be tokenized by WordPiece as:
  - [CLS] I ' m feeling fan ##tas ##tic ! [SEP]
- **Converting Tokens to IDs**: After tokenization, each token is mapped to an ID from BERT's vocabulary, which serves as input to the model.

BERT input encoding

The advantages of WordPiece embedding are the following:

- **Efficiency**: Reduces the vocabulary size without losing the ability to represent diverse language constructs.
- **Handling Unseen Words**: Subword tokenization allows BERT to manage rare or newly created words by breaking them down into recognizable parts.
- **Improved Language Understanding**: Helps BERT capture complex linguistic patterns by learning useful subword components during pretraining.

BERT [CLS] token

- In BERT, the [CLS] token plays a crucial role as a special token added at the beginning of each input sequence.
- The [CLS] token (short for "classification token") is always placed at the very start of the tokenized sequence and is primarily used as a summary representation of the entire input sequence.
- After the input is processed by BERT, the final hidden state of the [CLS] token acts as a condensed, **context-aware embedding** for the whole sentence or sequence of sentences.
- This embedding can then be fed into additional layers (like a classifier) for specific tasks.

BERT [CLS] token

The [CLS] token is used differently for single sentence and sentence pair classification.

- **Single-Sentence Classification**
  - The final hidden state of the [CLS] token is passed to a classifier layer to make predictions.
  - For instance, in sentiment analysis, the classifier might predict "positive" or "negative" sentiment based on the [CLS] embedding.

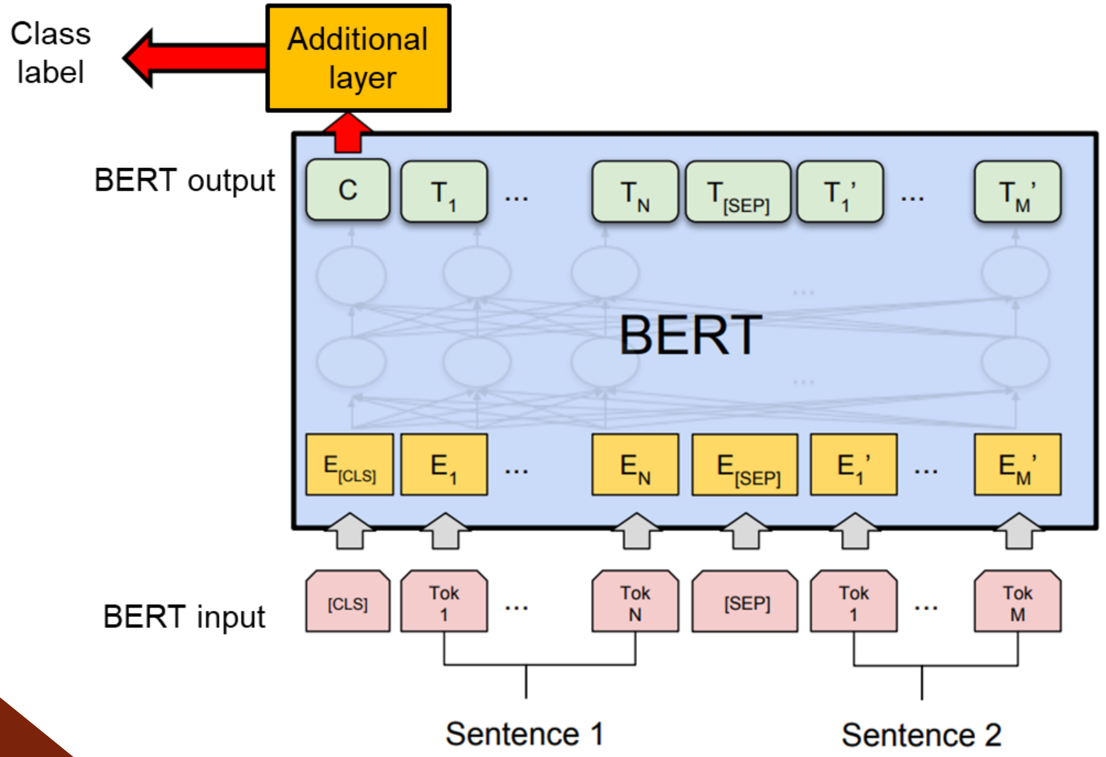
- **Sentence-Pair Tasks**
  - For tasks involving two sentences, BERT tokenizes them as [CLS] Sentence A [SEP] Sentence B [SEP].
  - The [CLS] token's final hidden state captures the relationship between the two sentences, making it suitable for tasks like entailment detection or similarity scoring.

BERT pre-training

- BERT is pre-trained using two self-supervised learning strategies:
  - **Masked Language Modeling (MLM)**
    - Objective: Predict masked (hidden) tokens in a sentence.
    - Process: Randomly mask 15% of tokens and train BERT to predict them based on context.
    - Benefit: Enables BERT to learn bidirectional context.
  - **Next Sentence Prediction (NSP)**
    - Objective: Determine if one sentence logically follows another.
    - Process: Trains BERT to understand sentence-level relationships.
    - Benefit: Improves performance in tasks like question answering and natural language inference.
- The training set is composed by a corpus of publicly available books plus the English Wikipedia, for a total of more than 3 billions of words

BERT fine tuning

- The output of the encoder is given to an additional layer to solve a specific problem
  - The cross-entropy loss between the prediction and the label for the classification task is minimized via gradient-based algorithms, where the additional layer is trained from scratch, while pretrained parameters of BERT may be updated or not



In this case (a sequence-to-single-value task) only the output for the start token is used

BERT fine tuning

- BERT is pretrained on general language data, then fine-tuned on specific datasets for each task.
- In fine-tuning, the [CLS] token’s final embedding is specifically trained for the downstream task, refining its ability to represent the input sequence in the way needed for that task.
- Example Tasks:
  - **Text Classification**: Sentiment analysis, spam detection.
  - **Named Entity Recognition (NER)**: Identifying names, dates, organizations, etc., within text.
  - **Question Answering**: Extractive QA where BERT locates answers within a passage.
- **Minimal Task-Specific Adjustments**: Only a few additional layers are added per task.

BERT strengths and limitations

- **Strengths**
  - **Bidirectional Contextual Understanding**: provides richer and more accurate representations of language.
  - **Flexibility in Transfer Learning**: BERT’s pretraining allows for easy adaptation to diverse NLP tasks.
  - **High Performance on Benchmark Datasets**: Consistently ranks at or near the top on datasets like SQuAD (QA) and GLUE (general language understanding).

- **Limitations**
  - **Large Model Size**: High computational and memory requirements make deployment challenging.
  - **Pretraining Costs**: Requires extensive computational resources, especially for BERT-Large.
  - **Fine-Tuning Time and Data**: Fine-tuning on new tasks requires labeled data and can still be time-intensive.

Popular BERT variants - RoBERTa

- **RoBERTa** is the acronym for Robustly Optimized BERT Approach and was developed by Facebook AI in 2019
- The main differences with respect to BERT are the following:
  - **Larger Training Corpus**: Trained on more data (200 billions) compared to BERT.
  - **Removed Next Sentence Prediction (NSP)**: Found that removing NSP improves performance.
  - **Longer Training and Larger Batches**: RoBERTa was trained for more iterations and used larger batches for robust language modeling.
  - **Dynamic Masking**: Masking is applied dynamically (i.e., different masks per epoch), leading to better generalization.
- RoBERTa consistently outperforms BERT on various NLP benchmarks, particularly in tasks requiring nuanced language understanding.

Popular BERT variants - ALBERT

- **ALBERT** is the acronym of A Lite BERT, developed by Google Research in 2019.
- The main differences with respect to BERT are:
  - **Parameter Reduction**: Uses factorized embedding parameterization to reduce model size.
  - **Cross-Layer Parameter Sharing**: Shares weights across layers to decrease the number of parameters.
  - **Sentence Order Prediction (SOP)**: Replaces NSP with SOP, which is better suited for capturing inter-sentence coherence.
- ALBERT achieves comparable results to BERT-Large with fewer parameters, making it more memory-efficient.
- ALBERT is faster and lighter, ideal for applications where resources are limited.

Popular BERT variants - DistilBERT

- **DistilBERT** is the acronym for Distilled BERT, developed by Hugging Face.
- The main differences with BERT are the following:
  - **Model Distillation**: Uses knowledge distillation to reduce BERT's size by about 40% while retaining 97% of its language understanding capabilities.
  - **Fewer Layers**: DistilBERT has 6 layers instead of 12 (for BERT-Base) but is optimized to perform similarly.
  - **Faster Inference**: Provides faster inference and lower memory usage, making it ideal for real-time applications.
- DistilBERT is widely used for lightweight applications that need a smaller and faster model without major accuracy trade-offs.

Popular BERT variants - TinyBERT

- **TinyBERT** was developed by Huawei
- The main differences with BERT are the following:
  - **Two-Step Knowledge Distillation**: Distills BERT both during pretraining and fine-tuning, further enhancing efficiency.
  - **Smaller and Faster**: TinyBERT is even smaller than DistilBERT, optimized for mobile and edge devices.
  - **Similar Accuracy**: Maintains accuracy close to that of BERT on various NLP tasks, especially when fine-tuned with task-specific data.
- TinyBERT is an ultra-compact version of BERT that is well-suited for resource-constrained environments.

Popular BERT variants - ELECTRA

- **ELECTRA** is the acronym for Efficiently Learning an Encoder that Classifies Token Replacements Accurately, developed by Google Research.
- Its differences with respect to BERT are:
  - **Replaced Token Detection**: Instead of masked language modeling, ELECTRA uses a generator-discriminator setup where the model learns to identify replaced tokens in text.
  - **Efficient Pretraining**: This approach allows ELECTRA to learn with fewer resources and converge faster than BERT.
  - **Higher Performance**: Often outperforms BERT on language understanding benchmarks with significantly less compute power.
- ELECTRA's training efficiency and robust performance make it appealing for applications where computational resources are limited.

Popular BERT variants - SciBERT

- **SciBERT** is tailored for applications in scientific literature, making it ideal for academic and research-oriented NLP.

- **Domain-Specific Pretraining**: Trained on a large corpus of scientific papers from domains like biomedical and computer science.
- **Vocabulary Tailored to Science**: Uses a vocabulary that better represents scientific terms and jargon.
- **Improved Performance on Scientific NLP Tasks**: Significantly outperforms BERT on tasks like scientific text classification, NER, and relation extraction.

Popular BERT variants - BioBERT

- **BioBERT** is widely adopted in the biomedical research field, aiding in information extraction and discovery from medical literature.

- **Biomedical Corpus**: Pretrained on a biomedical text corpus, including PubMed abstracts and PMC full-text articles.
- **Enhanced Performance on Biomedical Tasks**: Excels at biomedical-specific tasks such as medical NER, relation extraction, and question answering in healthcare.

Popular BERT variants - ClinicalBERT

- **ClinicalBERT** is ideal for hospitals and healthcare providers who need to analyze patient records, predict health outcomes, or assist in clinical decision-making.

- **Healthcare Focus**: Tailored for processing clinical notes and healthcare-related NLP tasks.
- **Training on MIMIC-III Dataset**: Pretrained on the MIMIC-III database of clinical records, making it useful for healthcare analytics.

Popular BERT variants - mBERT

- **mBERT**, developed by Google, supports NLP tasks across languages, enabling global applications and language transfer learning.

- **Multilingual Support**: Trained on 104 languages, mBERT can handle multilingual text without requiring separate models for each language.
- **Language-Agnostic Representation**: Capable of zero-shot cross-lingual transfer, making it suitable for translation and cross-lingual understanding tasks.

Other BERT variants

- **CamemBERT**: French language-focused BERT model.
- **FinBERT**: Optimized for financial text analysis.
- **LegalBERT**: Trained on legal documents for better performance in the legal domain.
- Moreover, BERT inspired Transformer pre-training in computer vision, such as with **vision Transformers**, **Swin Transformers**, and **Masked Auto Encoders** (MAE)

Practice on token classification and named entity recognition

Practice

- Looking at the Hugging Face tutorial on token classification https://huggingface.co/learn/nlp-course/chapter7/2?fw=pt , use different existing versions of BERT to perform named entity recognition
- Test these versions not only with your own prompts, but also with data available in public datasets (e.g. https://huggingface.co/datasets/eriktks/conll2003)
- If you have time and computational resources, you can also fine tune one of the lightweight versions of BERT (https://huggingface.co/learn/nlp-course/chapter7/2?fw=pt#fine-tuning-the-model-with-the-trainer-api)

Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica

Lesson 13

Encoder-only Transformers

Nicola Capuano and Antonio Greco

DIEM – University of Salerno