21

Advanced Transformers

Robots in disguise?

21.1
Positional Encoding.

Why do we need positional encoding

- In the context of Transformer models like BERT, positional encoding is essential because, unlike RNNs or LSTMs, Transformers process the input data in parallel and do not inherently capture the sequential nature of the data
- For instance, the word "bank" can have different meanings depending on its position and surrounding words in a sentence

How Positional Encoding Works

- There are different ways to implement positional encoding, but one common approach involves using sine and cosine functions of different frequencies
- For each position *i* and dimension *d* in the embedding space, a positional encoding vector P is defined
- This is very math heavy and should be studied more in-depth if you want to go down the path of research

Why Sinusoidal Functions

- For now let's understand why use sin and cos in the first place
  - **Predictability**: It allows the model to easily learn to attend by relative positions
  - **Scalability**: The sinusoidal pattern is not limited to a fixed length of the sequence
  - **Efficiency**: Using sinusoids allows for more efficient learning and easier extrapolation to longer sequence lengths

Alternatives to Sinusoidal methods

- Learned Positional Embeddings: Some models learn positional embeddings in the same way they learn word embeddings. This approach tailors the positional information more specifically to the dataset but may not generalize as well to sequence lengths outside the training data
- Relative Positional Encoding: This approach encodes the relative positions of tokens with respect to each other, rather than their absolute position in the sequence

21.2
Tokenization Methods.

Why explore Tokenization

- By itself, tokenization is an easy concept to understand
- However as it is a critical step in NLP, it is important to learn different methods of tokenization especially as it pertains to different tasks

Word Tokenization

- This is the process of splitting a piece of text into individual words based on certain delimiters like spaces, punctuation, etc.
- **Challenges**: The main challenge arises with languages that don't use spaces to separate words or with compound words in languages like German
- **Applications**: Basic NLP tasks like word frequency counts, bag-of-words model for text classification, etc.

Sentence Tokenization

- This method involves dividing a text into its constituent sentences. It usually relies on punctuation marks as indicators of sentence boundaries
- **Challenges**: Complexities arise with the use of punctuation in abbreviations, dates, or non-standard sentence endings
- **Applications**: Useful in tasks that require understanding context at the sentence level, like sentiment analysis, machine translation, and summarization

Byte Pair Encoding (BPE)

- BPE is a middle ground between word-level and character-level tokenization. It starts with a base vocabulary of individual characters and iteratively merges the most frequent pair of tokens
- **Challenges**: BPE may generate suboptimal splits of words if the training corpus isn't representative of the target text
- **Applications**: Efficient in representing common subword units, significantly reduces the vocabulary size, and handles out-of-vocabulary words well. Used in models like GPT-2, GPT-3

WordPiece

- Similar to BPE but chooses to merge tokens based on a likelihood of improving the language model's overall performance rather than frequency
- **Challenges**: Requires a more complex training process than simple BPE and might be sensitive to the training data
- **Applications**: Used in BERT and similar Transformer-based models. It helps these models handle a wide range of language phenomena

SentencePiece

- Unlike BPE and WordPiece, SentencePiece tokenizes text directly into subword units without relying on whitespace for initial word segmentation. This is crucial for languages where whitespace isn't a clear indicator of word boundaries
- **Challenges**: Might create overly fragmented tokens in languages with clear word boundaries marked by spaces

21.3
Transfer Learning.

Concept

- Transfer learning involves taking a model trained on a large dataset (often a general task like language modeling) and adapting it to a specific task
- This is particularly useful in NLP, where large models are trained on extensive text corpora and then adapted for specific tasks like sentiment analysis, question-answering, etc.

Benefits

- **Efficiency**: Reduces the computational cost and time as the model doesn’t need to be trained from scratch.
- **Performance**: Often leads to improved performance, especially in scenarios where the target task has limited data available.
- **Generalization**: Pre-trained models have generally learned rich representations that can be effectively transferred to new tasks.

Process

- **Pre-training**: A model is trained on a large dataset. In NLP, this could be a language model trained on a corpus like Wikipedia or web text
- **Adaptation**: The pre-trained model is then fine-tuned on a smaller, task-specific dataset. For example, a BERT model pre-trained on English text can be fine-tuned for a sentiment analysis task.

Fine-Tuning

- Fine-tuning is a specific form of transfer learning where the pre-trained model is slightly adjusted to adapt to a new, related task
- The key idea is to leverage the learned features and weight adjustments of the pre-trained model, adapting them to a new task with minimal additional training

Procedure

- **Weight Initialization**: Instead of starting from scratch, the model begins with weights from a pre-trained model.
- **Further Training**: The model is then trained (or fine-tuned) on a new dataset. This training is usually much shorter and requires less data than training a model from scratch
- **Adjustment of Learning Parameters**: Often, a lower learning rate is used, as major adjustments to the model weights could destroy the previously learned features

Applications in NLP

- **BERT and Similar Models**: These Transformer-based models are often fine-tuned for specific tasks like text classification, named entity recognition, etc. The process involves adding a task-specific layer if needed (like a classification layer) and training the model on a task-specific dataset
- Fine-tuning allows for customization of general models to specific domains or applications, even with small datasets

21.4
Multimodal Transformers.

What are Multimodal Transformers

- Multimodal Transformers are designed to process and relate information from different modalities (e.g., textual, visual, auditory)
- These models typically involve separate encoding for each modality before combining or fusing these representations. For instance, an image might be processed into a series of patches (as in Vision Transformers), and text might be tokenized and embedded

Fusion Mechanism

- Key to these models is the mechanism by which they fuse information from different modalities
- This can be done at various stages of the model (early, mid, or late fusion), and the approach can significantly impact the model's performance on multimodal tasks

Applications

- **Image and Text**: In tasks like image captioning or visual question answering, the model needs to understand and relate textual descriptions or questions to visual content
- **Video and Text**: Applications include video captioning, where the model generates descriptions of video content, and video question answering, where it answers questions based on video content

Existing Multimodal Transformers

- Models like ViLBERT and LXMERT extend the BERT architecture to handle both visual and textual inputs, training on tasks like visual question answering and image-text matching
- Audio-Visual Models: Projects like Audio-Visual Transformers (AVT) focus on synchronizing and interpreting audio-visual data, useful in tasks like audio-visual speech recognition

Challenges

- One of the key challenges is aligning features from different modalities
- This involves mapping the representations of different data types into a common space where they can be effectively compared and combined

Challenges

- Training multimodal Transformers can be challenging, especially in terms of data availability and balancing the influence of each modality.
- Techniques like transfer learning, where the model is pre-trained on unimodal tasks before multimodal training, can be beneficial

END.