Masked Language Modelling (MLM)

Bhupen Sinha

25-07-2024

TABLE OF CONTENTS

TRAINING OF A TRANSFORMER MODELS - OVERVIEW

Training transformer models involves several key steps and considerations:

1. **Data Preparation**:
    - **Dataset Collection**: Gather a large and diverse dataset relevant to your task (e.g., text for NLP tasks).
    - **Preprocessing**: Tokenize the text, handle special characters, convert text to numerical format, and create attention masks. For NLP, common tokenizers include BERT's WordPiece, GPT's Byte Pair Encoding (BPE), or SentencePiece.
2. **Model Architecture**:
    - **Transformers**: Understand the core components like Multi-Head Self-Attention, Feed-Forward Neural Networks, Positional Encoding, and Layer Normalization.
    - **Variants**: Choose the right variant of transformer architecture (e.g., BERT for bidirectional context, GPT for autoregressive tasks).
3. **Training Procedure**:
    - **Initialization**: Initialize model weights, usually with methods like Xavier or He initialization.
    - **Loss Function**: Define the loss function suitable for the task (e.g., Cross-Entropy Loss for classification, Mean Squared Error for regression).
    - **Optimizer**: Use an optimizer like Adam or AdamW with learning rate scheduling. Learning rate warm-up and decay can improve training stability.
    - **Training Loop**: Train the model over multiple epochs, with each epoch involving forward and backward passes, and weight updates. Monitor metrics such as loss and accuracy.
4. **Hyperparameter Tuning**:
    - **Batch Size**: Choose an appropriate batch size that fits your hardware capabilities.
    - **Learning Rate**: Adjust the learning rate and its schedule for optimal convergence.
    - **Number of Layers and Heads**: Experiment with different numbers of transformer layers and attention heads.
5. **Evaluation**:
    - **Validation**: Use a validation set to tune hyperparameters and avoid overfitting.
    - **Metrics**: Depending on the task, evaluate the model using appropriate metrics (e.g., accuracy, F1 score, BLEU score).

6. **Fine-Tuning**:
    - **Pre-trained Models**: Start with pre-trained transformer models and fine-tune them on your specific dataset to leverage existing knowledge and reduce training time.
7. **Regularization and Optimization**:
    - **Dropout**: Apply dropout to prevent overfitting.
    - **Gradient Clipping**: Use gradient clipping to manage exploding gradients.
8. **Inference and Deployment**:
    - **Model Export**: Save the trained model for inference.
    - **Optimization for Deployment**: Convert the model to a format suitable for deployment (e.g., ONNX or TensorRT) and optimize for inference speed.

Training transformer models requires significant computational resources and careful tuning of hyperparameters to achieve optimal performance.

TRAINING METHODS FOR TRANSFORMERS

1. **Supervised Learning**:
    - **Classification**: Training on labeled data where the model learns to predict class labels (e.g., text classification, sentiment analysis).
    - **Regression**: Training on labeled data where the model learns to predict continuous values (e.g., predicting numeric values from text).
2. **Unsupervised Learning**:
    - **Language Modeling**: Training on large corpora of text to predict the next word in a sequence (e.g., GPT, BERT). This includes:
        - **Causal Language Modeling**: Predicting the next token in a sequence (used in autoregressive models like GPT).
        - **Masked Language Modelling**: Predicting missing tokens in a sequence (used in bidirectional models like BERT).
3. **Self-Supervised Learning**:
    - **Contrastive Learning**: Learning to differentiate between similar and dissimilar examples, often using techniques like SimCLR or MoCo. In transformers, this might involve learning representations that are invariant to certain transformations.
    - **Denoising Autoencoders**: Training models to reconstruct corrupted or masked input data. For example, BERT uses a masked language model objective where some tokens are randomly masked, and the model learns to predict them.
4. **Transfer Learning**:
    - **Pre-training and Fine-Tuning**: First, a transformer model is pre-trained on a large, generic dataset (e.g., BERT pre-trained on Wikipedia), and then it is fine-tuned on a specific task with task-specific data.
5. **Multi-Task Learning**:
    - **Joint Training**: Training the model on multiple related tasks simultaneously (e.g., training a model to perform both question answering and sentiment analysis). This can help the model generalize better across tasks.
6. **Reinforcement Learning**:
    - **Reward-Based Fine-Tuning**: Using reinforcement learning to fine-tune a model based on rewards, often applied in tasks like dialogue systems where the quality of generated responses can be evaluated with rewards (e.g., RLHF - Reinforcement Learning from Human Feedback).
7. **Meta-Learning**:

- **Learning to Learn**: Training models to quickly adapt to new tasks or domains with minimal data. This is less common but involves techniques like MAML (Model-Agnostic Meta-Learning).

8. **Few-Shot and Zero-Shot Learning**:
    - **Prompt-Based Learning**: Leveraging pre-trained transformers to perform tasks with minimal examples or even without explicit training examples (e.g., GPT-3's few-shot and zero-shot learning capabilities).

MASKED LANGUAGE MODELLING - OVERVIEW

Masked Language Modelling (MLM) is a self-supervised learning approach used primarily for training bidirectional transformer models like BERT.

Here's a detailed description of how MLM works:

Concept

Objective: The main goal of MLM is to train a model to predict missing or masked tokens in a sequence. This helps the model learn rich, contextual representations of words by understanding the relationships between them within a given text.

Process

1. **Masking Tokens**:
    - During training, a certain percentage of the tokens in the input sequence are randomly replaced with a special [MASK] token. Typically, around 15% of the tokens are masked.
    - The choice of which tokens to mask is randomized, ensuring that the model does not learn to rely on the positions of the masks but instead learns contextual information from surrounding tokens.
2. **Model Training**:
    - The model is then trained to predict the original tokens for the masked positions based on the context provided by the unmasked tokens.
    - For instance, in the sentence "The cat sat on the [MASK]," the model should predict "mat" for the masked token.
    - The model processes the entire sequence bidirectionally, meaning it takes into account both the preceding and succeeding tokens to make predictions. This bidirectional approach allows the model to understand the context from both directions, leading to more accurate and contextually aware representations.
3. **Loss Calculation**:
    - The model's predictions are compared against the actual original tokens using a loss function, typically cross-entropy loss. This loss quantifies the difference between the predicted probabilities and the actual token distribution.
    - The model's weights are updated through backpropagation to minimize this loss, refining the model's ability to predict masked tokens based on context.
4. **Model Representation**:
    - Through the MLM objective, the model learns to capture complex relationships and contextual dependencies between tokens. This enables the model to generate rich, contextual embeddings for each token in a sentence, which are useful for various downstream tasks.

Masked Language Modelling (MLM)

Benefits

- **Bidirectional Context**: Unlike unidirectional models (e.g., GPT, which only considers tokens to the left of a given position), MLM allows the model to use both preceding and succeeding tokens, providing a more comprehensive understanding of context.
- **Rich Representations**: By learning to predict masked tokens, the model develops nuanced representations of words that capture their meanings in various contexts.
- **Versatility**: The contextual embeddings learned through MLM can be fine-tuned for a wide range of NLP tasks, such as text classification, named entity recognition, and question answering.

Example

Given the sentence: "The quick brown fox jumps over the [MASK] dog."

- **Training Input**: "The quick brown fox jumps over the [MASK] dog."
- **Training Target**: "The quick brown fox jumps over the lazy dog."

The model learns to predict that the masked token [MASK] should be "lazy" based on the context provided by the surrounding words.

MASKING STRATEGIES

1. **Random Masking**:
    - **Description**: Randomly select a subset of tokens in the input sequence to replace with the [MASK] token. This is the most common and straightforward approach.
    - **Implementation**:
        - Typically, around 15% of the tokens are chosen randomly to be masked.
        - Of the selected tokens, 80% are replaced with [MASK], 10% are replaced with a random token, and 10% are left unchanged (but the model is still trained to predict them). This variation prevents the model from overfitting to the [MASK] token and encourages it to rely on contextual information.
    - **Example 1**:
        - Original: "The quick brown fox jumps over the lazy dog."
        - Masked: "The quick [MASK] fox jumps [MASK] the lazy dog."
    - **Example 2**:
        - Original: "The cat sat on the mat."
        - Masked: "The [MASK] sat on the mat."
    - **Example 3**:
        - Original: "She loves reading books."
        - Masked: "She [MASK] reading books."
2. **Dynamic Masking**:
    - **Description**: Masking strategies are applied dynamically during training, meaning the specific tokens that are masked can change in each training epoch or iteration. This approach introduces more variability and ensures that the model sees a broader range of contexts during training.
    - **Implementation**:
        - Masking patterns are generated on-the-fly during each training pass, so the same token may not be masked in every epoch.
        - Helps prevent overfitting by exposing the model to different masked contexts over time.

- **Example**:

  (Epoch 1)

  - **Original**: "The quick brown fox jumps over the lazy dog."
  - **Masked**: "The quick [MASK] fox jumps over the lazy dog."

  (Epoch 2):

  - **Original**: "The quick brown fox jumps over the lazy dog."
  - **Masked**: "The [MASK] brown fox [MASK] over the lazy dog."

3. **Span-Based Masking**:

- **Description**: Mask contiguous spans of tokens rather than individual tokens. This can help the model learn dependencies over longer contexts.
- **Implementation**:
  - Randomly select spans of varying lengths (e.g., 1-5 tokens) to mask within the sequence.
  - Especially useful for tasks where understanding the relationship between adjacent tokens is crucial.
- **Example**:
  - Original: "The quick brown fox jumps over the lazy dog."
  - Masked: "The [MASK] [MASK] fox jumps [MASK] the lazy dog."

4. **Whole Word Masking**:

- **Description**: When masking, entire words are masked rather than subword tokens. This approach is particularly useful for languages or tokenization schemes that break words into subword units.
- **Implementation**:
  - Ensure that when a part of a word is chosen to be masked, the entire word is masked.
  - Helps the model learn more holistic representations of words.
- **Example**:
  - Original: "The quick brown fox jumps over the lazy dog."
  - Masked: "The quick [MASK] [MASK] jumps over the lazy dog."

5. **Entity-Based Masking**:
    - **Description**: Specifically target entities or meaningful phrases for masking, such as names, dates, or locations.
    - **Implementation**:
        - Use named entity recognition (NER) to identify entities in the text.
        - Randomly mask these identified entities during training.
    - **Example**:
        - Original: "Barack Obama was born in Hawaii."
        - Masked: "[MASK] [MASK] was born in [MASK]."

6. **Mixture of Masking Techniques**:
    - **Description**: Combine multiple masking techniques within a single training process to leverage the benefits of each.
    - **Implementation**:
        - Alternate or mix different masking strategies (e.g., random, span-based, syntax-aware) during training.
        - Adjust the proportion of each technique based on the training progress or task requirements.
    - **Example**: Using random masking for one batch and span-based masking for the next.

7. **Frequency-Based Masking**:
    - **Description**: Mask tokens based on their frequency in the training corpus, focusing on either high-frequency (common) or low-frequency (rare) tokens.
    - **Implementation**:
        - Calculate token frequencies in the training data.
        - Mask tokens based on predefined frequency thresholds or distributions.
    - **Example**: Masking rare words to encourage the model to learn from infrequent context or common words to ensure they are well-represented.

KEY POINTERS ON VARIOUS MASKING TECHNIQUES

Masking Technique	Description	Impacts
Random Masking	Randomly selects tokens to mask.	- Simple and easy to implement. - Encourages the model to generalize. - Risk of overfitting to specific masked tokens if not varied enough.
Dynamic Masking	Applies masking strategies dynamically, changing in each epoch.	- Increases model robustness - varied masked contexts. - Helps prevent overfitting. - Requires more computational resources.
Span-Based Masking	Masks contiguous spans of tokens.	- Captures dependencies over longer contexts. - Better for understanding phrases and multi-token entities. - Can be more complex to implement.
Whole Word Masking	Masks entire words rather than subword tokens.	- Helps the model learn holistic word representations. - More effective for languages with rich morphology. - Slightly higher computational complexity.
Entity-Based Masking	Masks named entities or meaningful phrases.	- Enhances understanding of important entities. - Useful for tasks requiring entity recognition. - Requires entity recognition pre-processing.
Adaptive Masking	Adjusts masking probability based on token informativeness.	- Focuses learning on more informative tokens. - Potentially improves learning efficiency. - Requires a scoring mechanism for informativeness.
Syntax-Aware Masking	Uses syntactic information to guide masking.	- Targets syntactically significant tokens. - Helps the model learn syntactic structures. - Requires syntactic parsing.
Contextual Masking	Masks tokens based on their context within the sentence.	- Focuses on contextually unique or central tokens. - improves contextual understanding. - Requires contextual analysis.
Semantic Masking	Uses semantic information to guide masking.	- Targets semantically rich tokens. - Enhances semantic understanding. - Requires semantic analysis.
Mixture of Masking Techniques	Combines multiple masking strategies.	- Leverages benefits of multiple strategies. - Increases robustness and generalization. - More complex implementation and tuning required.
Frequency-Based Masking	Masks tokens based on their frequency in the corpus.	- Balances learning between common and rare words. - generalize better across different word frequencies. - Requires frequency analysis.

MASKING STRATEGIES FOR SPECIFIC NLP APPLICATION

Masking Technique	Suitable for NLP Applications	Explanations
Random Masking	General NLP tasks, such as text classification and sentiment analysis	Provides a broad context learning mechanism that helps in generalizing across various tasks.
Dynamic Masking	Large-scale pretraining, diverse and dynamic datasets	Exposes the model to varied contexts, improving robustness and preventing overfitting.
Span-Based Masking	Named Entity Recognition (NER), question answering	Captures dependencies over longer contexts, beneficial for understanding phrases and entities.
Whole Word Masking	Language modeling, machine translation	Helps in learning complete word representations, useful for tasks requiring rich morphological understanding.
Entity-Based Masking	NER, information extraction	Focuses on entities, improving the model's ability to understand and recognize important entities.
Adaptive Masking	Specialized domain tasks, adaptive learning scenarios	Improves learning efficiency by focusing on informative tokens, suitable for specialized domains.
Syntax-Aware Masking	Syntax-based tasks, syntactic parsing	Targets syntactically significant tokens, aiding in learning syntactic structures.
Contextual Masking	Context-sensitive tasks, semantic similarity, contextual embeddings	Enhances contextual understanding by focusing on central tokens within sentences.
Semantic Masking	Semantic analysis, sentiment analysis, semantic search	Focuses on semantically rich tokens, improving semantic understanding and analysis.
Mixture of Masking Techniques	General NLP pretraining, multitask learning	Combines benefits of multiple strategies, increasing robustness and generalization.
Frequency-Based Masking	Language modeling, rare word recognition	Balances learning between common and rare words, enhancing generalization across word frequencies.

DIVE INTO A FEW MASKING TECHNIQUES

1. RANDOM MASKING

Transformer Models:

- **BERT (Bidirectional Encoder Representations from Transformers)**

Datasets:

- **BooksCorpus**: A large dataset consisting of over 11,000 unpublished books.
- **English Wikipedia**: The entire English Wikipedia dump.

Key Notes/Points:

- 15% of the tokens in the input sequence are selected for masking.
- Out of the selected tokens, 80% are replaced with the [MASK] token, 10% are replaced with a random token, and 10% remain unchanged.
- This strategy helps in creating a diverse set of training examples, aiding the model in generalizing better across different contexts.

Explanation:

BERT uses random masking to predict missing words in sentences.

This method ensures that the model learns to understand context and predict the correct tokens even when some tokens are masked.

The varied replacement strategy (80/10/10) adds randomness and prevents the model from overfitting to specific masked patterns.

2. DYNAMIC MASKING

Transformer Models:

- **RoBERTa (Robustly Optimized BERT Approach)**

Datasets:

- **Common Crawl News**: A dataset containing news articles.
- **OpenWebText**: A dataset based on the web pages found in the open web.
- **BooksCorpus**: A large dataset consisting of over 11,000 unpublished books.
- **Stories from Common Crawl**: A dataset with story-like text extracted from Common Crawl.

Key Notes/Points:

- Masking patterns change dynamically across epochs, meaning the same token may be masked differently in different epochs.
- Helps in improving model robustness and reducing the risk of overfitting.
- Requires more computational resources due to the dynamic nature of masking.

Explanation:

RoBERTa employs dynamic masking, where the same token may be masked in different ways in different training epochs.

This prevents the model from memorizing the position of masked tokens and enhances its ability to generalize to new data by providing a more varied learning experience.

3. SPAN-BASED MASKING

Transformer Models:

- **SpanBERT (Span-based BERT)**

Datasets:

- **BooksCorpus**: A large dataset consisting of over 11,000 unpublished books.
- **English Wikipedia**: The entire English Wikipedia dump.

Key Notes/Points:

- Instead of masking individual tokens, contiguous spans of tokens are masked.
- This method captures dependencies over longer contexts, which is particularly useful for tasks involving multi-token entities and phrases.
- Helps the model in learning more about the relationships between words within a span.

Explanation:

SpanBERT masks spans of tokens rather than individual tokens.

This helps the model learn the relationships between words in a contiguous segment, making it more effective for tasks that involve understanding phrases or multiple tokens together, such as named entity recognition and question answering.

4. WHOLE WORD MASKING

Transformer Models:

- **Whole Word Masked Language Model (WWMLM)**

Datasets:

- **BooksCorpus**: A large dataset consisting of over 11,000 unpublished books.
- **English Wikipedia**: The entire English Wikipedia dump.

Key Notes/Points:

- Whole words are masked instead of subword tokens.
- Useful for languages with rich morphology as it helps in learning coherent word representations.
- Ensures that the model predicts entire words rather than parts of words, leading to better word-level understanding.

Explanation:

In whole word masking, entire words are masked rather than their subword components. This ensures that the model learns to predict full words, which is beneficial for tasks that require understanding complete word forms and meanings, especially in languages with complex morphology.

5. ENTITY-BASED MASKING

Transformer Models:

- **ERNIE (Enhanced Representation through Knowledge Integration)**

Datasets:

- **Knowledge Graphs**: Structured datasets containing entity relationships.
- **Wikipedia**: The entire Wikipedia dump.

Key Notes/Points:

- Named entities and meaningful phrases are masked.
- Focuses on improving the model's understanding of entities, crucial for tasks like named entity recognition and information extraction.
- By learning to predict masked entities, the model gains a better understanding of key information and relationships.

Explanation:

ERNIE masks named entities and significant phrases, enhancing the model's ability to understand and predict important entities within the text.

This is particularly useful for tasks that involve recognizing and working with specific entities, such as named entity recognition and information extraction.

USEFUL LINKS AND REFERENCES

-
-

INDEX

No index entries found.