Model Selection and Traning

Model Selection and Training Documentation

Model Selection

1. Primary Model: PubMedBERT

We selected the `microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract-fulltext` model as our primary embedding model for the following reasons:

Domain Specificity

- Pre-trained on biomedical and clinical text from PubMed abstracts and full-text articles
- Better understanding of medical terminology and concepts compared to general-purpose language models
- Demonstrated superior performance on biomedical NLP tasks in various benchmarks

Architecture Benefits

- Based on BERT architecture (12 layers, 768 hidden size, 12 attention heads)
- Uncased model reduces vocabulary complexity while maintaining performance
- 512 token length limit suitable for clinical trial descriptions

Performance Characteristics

- Memory efficient (base model vs large)
- Good balance between accuracy and computational requirements
- Supports efficient batching and GPU acceleration

2. Alternative Models Considered

1. Clinical BERT

- **Why Not Selected**: While domain-specific, showed slower inference times and higher memory usage

2. Sentence-BERT Models

- **Why Not Selected**: Generic models lacked domain expertise, though faster inference

3. BioBERT

- **Why Not Selected**: Comparable performance to PubMedBERT but larger model size

Model Configuration & Parameters

1. Tokenizer Settings

- Max length: 512 tokens
- Padding: True (dynamic padding per batch)
- Truncation: True
- Return tensors: PyTorch tensors

2. Model Parameters

- Mixed precision training (FP16)
- Batch size: 32 (optimized for RTX 3090)
- Mean pooling for sentence embeddings

Training Process

1. Hardware Environment

- GPU: NVIDIA RTX 3090 (24GB VRAM)
- RAM: 125GB
- CUDA Version: 11.8.0
- System: Ubuntu 22.04

2. Software Environment

- Python 3.10
- PyTorch 2.1.0
- Transformers 4.37.2
- Additional requirements:

```
name: clinical-trials
channels:
  - pytorch
  - conda-forge
  - defaults
dependencies:
  - python=3.10
  - pytorch=2.1.0
  - transformers=4.37.2
  - scikit-learn=1.3.2
  - pandas
  - numpy
  - tqdm
  - spacy
  - textblob
  - gdown
  - pip
```

```
  - pip:
    - torch==2.1.0
    - nltk==3.8.1
```

3. Optimization Techniques

Memory Optimization

- Gradient checkpointing enabled
- Mixed precision training (FP16)
- Batch size optimization
- Regular GPU memory clearing

```
# Memory optimization code
torch.backends.cuda.matmul.allow_tf32 = True
torch.backends.cudnn.allow_tf32 = True

if torch.cuda.is_available():
    self.model.half()  # Use FP16
    torch.cuda.empty_cache()
```

Performance Optimization

- Dynamic batching based on sequence length
- Efficient data loading using chunking
- Text preprocessing optimization

```
# Batch optimization
def encode_texts(self, texts, batch_size=32):
    embeddings = []
    for i in tqdm(range(0, len(texts), batch_size), desc="Encoding"):
        batch = texts[i:i + batch_size]
        encoded = self.tokenizer(
            batch,
            padding=True,
            truncation=True,
            max_length=512,
            return_tensors='pt'
        )
```

4. Hyperparameter Selection

Text Processing Weights

Optimized through experimentation:

- Study Title: 3.0

- Primary Outcome Measures: 2.5
- Secondary Outcome Measures: 2.0
- Criteria: 2.5
- Conditions: 2.5
- Interventions: 2.0
- Phase: 1.5
- Brief Summary: 1.8

Embedding Generation

- Batch size: 32 (optimal for RTX 3090)
- Max sequence length: 512
- Learning rate: 2e-5
- Weight decay: 0.01

Reproducibility

1. Random Seed Settings

All random seeds are set for reproducibility:

```
import numpy as np
import torch
import random

def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
```

2. Environment Setup

1. Create conda environment:
    - conda env create -f environment.yml
2. Activate environment:
    - conda activate clinical-trials
3. Install additional dependencies:
    - python -m spacy download en_core_web_sm

3. Dataset Preparation

- Use provided data loading scripts
- Maintain consistent preprocessing steps
- Follow documented cleaning procedures

Performance Metrics

1. Resource Usage

- GPU Memory: Peak usage ~18GB
- CPU Memory: Peak usage ~80GB
- Processing Time: ~25-30 minutes for full dataset

2. Model Performance

- Average Embedding Generation Speed: ~1000 trials/minute
- Similarity Computation Speed: ~5000 comparisons/second
- Memory Efficiency: ~75MB per 1000 trials

Validation Results

1. Similarity Metrics

- Cosine similarity distribution:
  - Mean: 0.85
  - Median: 0.87
  - Standard Deviation: 0.08

2. Cross-Validation

- 5-fold cross-validation on similarity scores
- Average precision at k=10: 0.92
- Average recall at k=10: 0.85

Future Improvements

1. Model Enhancements:
   - Experiment with newer biomedical transformers
   - Implement attention pooling
   - Test domain adaptation techniques
2. Performance Optimization:
   - Implement quantization for inference
   - Explore distributed computing
   - Optimize memory usage further

Conclusion

The selected model and configuration provide a robust foundation for clinical trial similarity analysis. The combination of PubMedBERT with optimized parameters and careful implementation choices ensures both accuracy and computational efficiency. All steps are reproducible following the provided documentation and environment settings.