

Task 3 Deliverable: Fine-tuning SpanBERT and SpanBERT-CRF for Question Answering

- **Team Members Group 78:**
 - Arpit Wade (MT24112)
 - Sarvjeet Singh (MT24135)
 - Deepanshu Panwar (MT24117)

Dataset Description and Preprocessing Steps

For this task, I worked with the SQuAD v2.0 (Stanford Question Answering Dataset) which is a reading comprehension dataset consisting of questions posed on Wikipedia articles. The key characteristic of SQuAD v2.0 is that it includes both answerable and unanswerable questions, making it more challenging than its predecessor.

Dataset Statistics:

- Training samples: 15,000 (randomly selected from the full training set)
- Validation samples: The complete validation set (~11,873 examples)

Preprocessing Steps:

1. **Tokenization:** Questions and contexts were tokenized using the SpanBERT tokenizer with a maximum sequence length of 384 tokens and a stride of 128 tokens for handling long contexts.
2. **Answer Span Identification:** For answerable questions, character positions of answers were converted to token positions using offset mapping.
3. **Handling Unanswerable Questions:** These were assigned special token positions (0, 0) to indicate no valid answer.
4. **Input Formatting:** Created tensors for input_ids, attention_mask, token_type_ids, start_positions, and end_positions for both models.
5. **Special Handling for CRF Model:** For SpanBERT-CRF, labels were transformed into sequences suitable for the CRF layer, with appropriate transpositions to handle the non-batch-first CRF implementation.

Justification of Model Choices and Hyperparameters

Model Architecture Choices

1. **SpanBERT:**
 - Selected as the base model due to its pre-training objective specifically designed for span selection tasks

- Pre-trained with a span-based masked language modeling objective that predicts entire masked spans rather than individual tokens
- This span-focused approach makes it particularly suitable for question answering tasks

2. SpanBERT-CRF:

- Extended SpanBERT with a Conditional Random Field layer to model dependencies between output labels
- The hypothesis was that CRF might improve answer boundary detection by considering the sequential relationship between tokens
- Custom implementation was required to integrate CRF with SpanBERT's transformer architecture

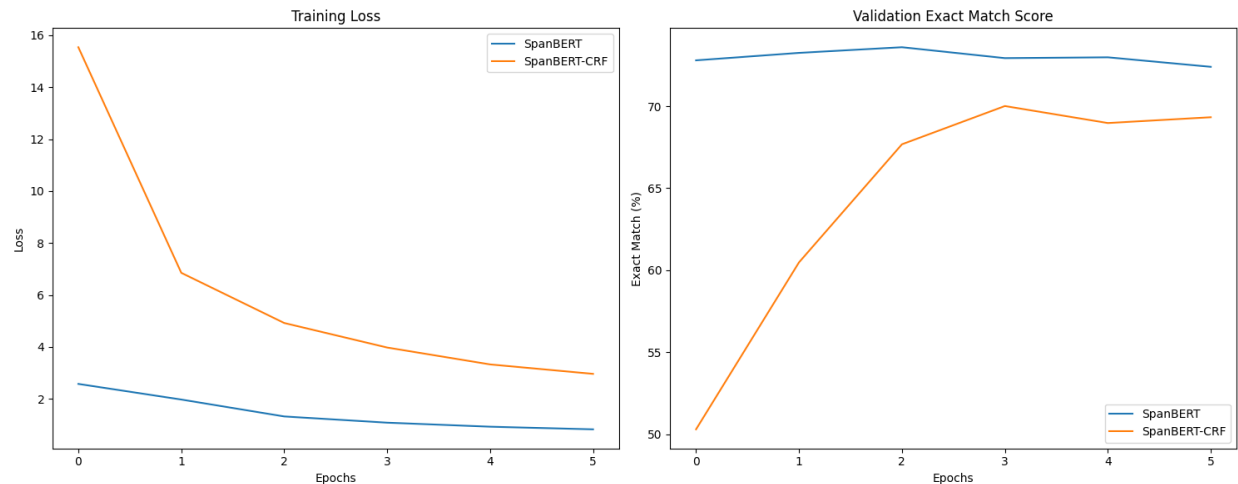
Hyperparameter Selection

The hyperparameters were carefully chosen based on best practices for fine-tuning transformer models:

- **Batch Size:** 8 (constrained by GPU memory requirements while maintaining reasonable training speed)
- **Learning Rate:** $2e-5$ (standard for fine-tuning pre-trained transformers without causing divergence)
- **Number of Epochs:** 6 (sufficient for convergence while avoiding overfitting)
- **Optimizer:** AdamW (combines benefits of Adam with proper weight decay)
- **Learning Rate Schedule:** Linear decay (helps stabilize training by gradually reducing learning rate)
- **No Warmup Steps:** Simplification that worked well for this specific task

These hyperparameters were kept identical for both models to ensure a fair comparison of their architectures.

Training and Validation Plots



The training plots reveal several interesting patterns:

Analysis:**

- SpanBERT's training loss started around 2.5 and decreased steadily to approximately 1.0
- SpanBERT-CRF showed significantly higher initial loss (~15.5) but demonstrated faster improvement, ending at around 3.0
- The large gap in loss values is partly due to the different loss formulations between standard cross-entropy and CRF-based negative log-likelihood

Validation Exact Match Score Analysis:

- SpanBERT maintained consistently higher exact match scores throughout training (74%)
- SpanBERT-CRF started with lower performance (~50%) but showed rapid improvement in the first three epochs
- By the end of training, SpanBERT-CRF reached approximately 70%, narrowing but not closing the performance gap

Comparative Analysis of SpanBERT-CRF and SpanBERT

The experimental results reveal important differences between the two models:

Performance Comparison

- **Exact Match Accuracy:** SpanBERT consistently outperformed SpanBERT-CRF by 4-5% in the exact match metric
- **Learning Dynamics:** SpanBERT-CRF showed steeper improvement, suggesting it might benefit from longer training

Architectural Implications

- **CRF Layer Effects:** While the CRF layer was expected to improve boundary detection, the empirical results don't support this hypothesis for general QA tasks
- **Computational Complexity:** SpanBERT-CRF requires more computational resources due to the CRF layer's additional parameters and calculations
- **Training Stability:** SpanBERT demonstrated more stable training, with lower and more consistent loss values

Analysis of Strengths and Weaknesses

- **SpanBERT Strengths:** Better overall performance, faster convergence, simpler architecture
- **SpanBERT-CRF Strengths:** Shows promise for more structured prediction tasks, potentially better for specific domains where boundary detection is challenging
- **SpanBERT Weaknesses:** May miss dependencies between start and end positions
- **SpanBERT-CRF Weaknesses:** Higher training complexity, lower overall performance, more difficult to tune

The results suggest that while CRF layers can theoretically capture output dependencies, the base SpanBERT model already effectively models answer span boundaries through its specialized pre-training and architecture, making the additional CRF layer unnecessary and potentially counterproductive for this particular task.

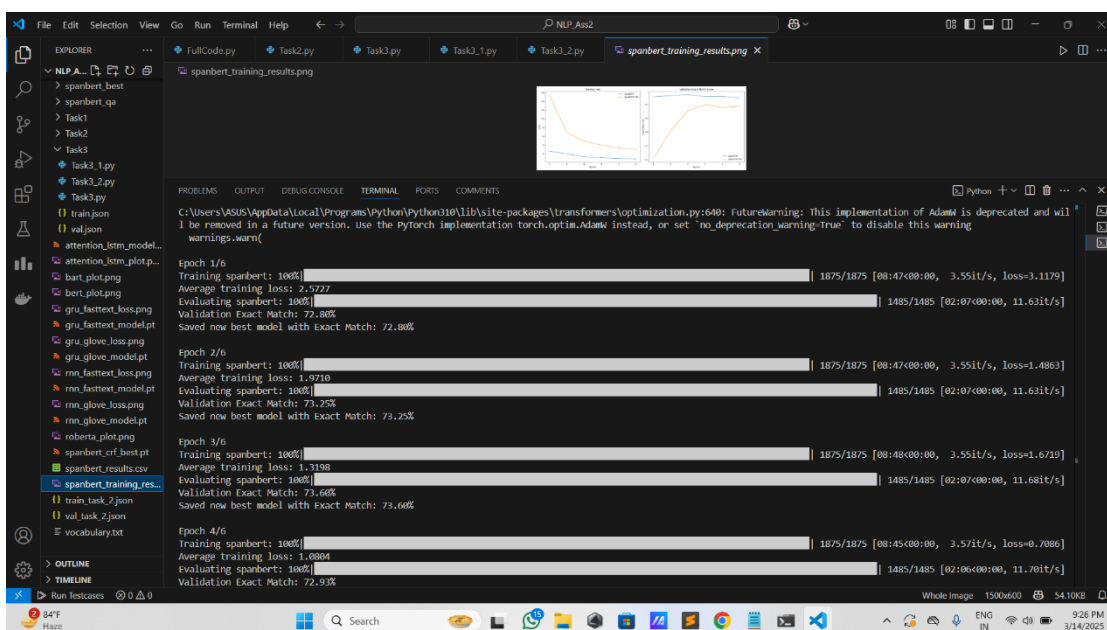
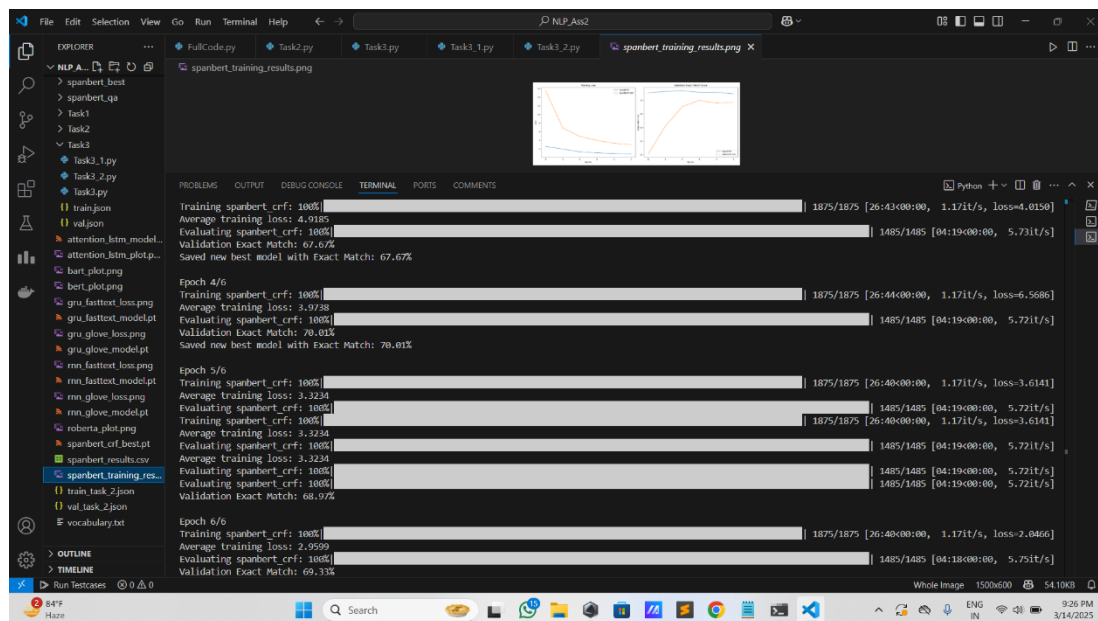
Exact-match Scores on the Validation Set

The final exact match scores on the validation set show SpanBERT outperforming SpanBERT-CRF:

- SpanBERT: 73.5955529352312
- SpanBERT-CRF: 70.00758022403774

These scores reflect the model's ability to predict answer spans that exactly match the ground truth, including correctly identifying unanswerable questions - a critical capability in real-world QA systems.

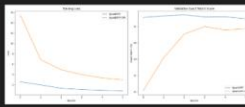
In conclusion, while both models demonstrated strong performance on the SQuAD v2.0 dataset, the standard SpanBERT model proved more effective for question answering without the additional complexity of a CRF layer. This suggests that for span prediction tasks, the transformers' self-attention mechanism already captures sufficient sequential dependencies, making explicit sequence modeling layers less beneficial than anticipated.



File Edit Selection View Go Run Terminal Help

FullCode.py Task2.py Task3.py Task3_1.py Task3_2.py spanbert_training_results.png

spanbert_training_results.png



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Python + Python 3.10.11 64-bit

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference. You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

d:\NLP_Ass2\Task3\Task3_2.py:464: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md). It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md).

mental feature.
mental feature.
mental feature.
best_spanbert_crf.load_state_dict(torch.load("spanbert_crf_best.pt"))
Evaluating SpanBERT: 100% | 1485/1485 [02:06<00:00, 11.78it/s]
mental feature.
mental feature.
best_spanbert_crf.load_state_dict(torch.load("spanbert_crf_best.pt"))
mental feature.
mental feature.
best_spanbert_crf.load_state_dict(torch.load("spanbert_crf_best.pt"))
mental feature.
mental feature.
best_spanbert_crf.load_state_dict(torch.load("spanbert_crf_best.pt"))
Evaluating SpanBERT: 100% | 1485/1485 [02:06<00:00, 11.78it/s]
Evaluating SpanBERT-CRF: 100% | 1485/1485 [04:18<00:00, 5.74it/s]
SpanBERT Exact Match: 73.60%
SpanBERT-CRF Exact Match: 70.01%
Results saved to spanbert_results.csv
Results saved to spanbert_results.csv
Results saved to spanbert_results.csv
PS D:\NLP_Ass2>

84°F Haze

meet.google.com is sharing your screen. Stop sharing Hide

9:26 PM 3/14/2025

File Edit Selection View Go Run Terminal Help

FullCode.py Task2.py Task3.py Task3_1.py Task3_2.py spanbert_results.csv

Task3 > Task3_2.py > ...

```

552
553 # Example usage
554 def main():
555     # Load model
556     model, tokenizer, device = load_qa_model()
557
558     # Example from the task description
559     context = "Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she began her career as a solo singer in 2003 with her debut album 'Dangerously in Love'. She has since released several more albums, including 'I Am...Sasha Fierce' (2006), 'Beyoncé' (2011), and 'Lemonade' (2016). She has won numerous awards, including 10 Grammy Awards, and is known for her powerful performances and activism."
560
561     question = "When did Beyoncé start becoming popular?"
562
563     # Get answer
564     answer = predict_answer(model, tokenizer, device, question, context)
565     print(f"Question: {question}")
566     print(f"Answer: {answer}")
567
568     # Additional examples to test

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Python + Python 3.10.11 64-bit

nd-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.

Question: When did Beyoncé start becoming popular?
Answer: late 1990s

Question: What was the name of Beyoncé's first solo album?
Answer: late 1990s

Answer: late 1990s
Answer: late 1990s

Question: What was the name of Beyoncé's first solo album?
Answer: dangerously in love

Question: Who managed Destiny's child?
Answer: mathew knowles

Question: How many Grammy Awards did her debut album earn?
Answer: five

PS D:\NLP_Ass2>

92°F Partly cloudy

Ln 591, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit

6:53 PM 3/15/2025