AI Tools Assignment
Date: June 09, 2025
Theoretical Understanding and Ethics and Optimization

---

## Part 1: Theoretical Understanding

1.

Q1: Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

TensorFlow and PyTorch are leading deep learning frameworks with distinct characteristics:

- Computational Graph:
    - TensorFlow uses a static computational graph  where the graph is built and optimized before execution, ensuring efficiency in production environments.
    - PyTorch employs a dynamic computational graph allowing real-time modifications which simplifies debugging and experimentation.
- Ease of Use
    - TensorFlow's high-level Keras API is user-friendly but can feel less flexible for complex models. It requires understanding graph-based operations.
    - PyTorch's Pythonic syntax is intuitive  making it ideal for rapid prototyping and research.
- Deployment
    - TensorFlow excels in production with tools like TensorFlow Serving for scalable deployment and TensorFlow Lite for mobile/edge devices.
    - PyTorch requires conversion to ONNX or TorchScript for similar deployment, which can add complexity.
- Community and Ecosystem
    - TensorFlow has broader industry adoption and a robust ecosystem.
    - PyTorch dominates in academic research due to its flexibility and active research community.

When to Choose:

- TensorFlow: Choose for production-grade systems, mobile/edge deployment, or when leveraging pre-trained models from TensorFlow Hub.
- PyTorch: For research, rapid prototyping, or tasks requiring dynamic graphs.

Q2: Describe two use cases for Jupyter Notebooks in AI development.

- Exploratory Data Analysis
    - Jupyter Notebooks enable interactive data exploration with libraries like Matplotlib, Seaborn, and Pandas. Developers can visualize datasets to identify patterns correlations before model development.
- Model Prototyping and Debugging:
    - Notebooks allow iterative model development, where developers can test architectures, adjust hyperparameters, and visualize metrics  in real-time streamlining experimentation and debugging.

Q3: How does spaCy enhance NLP tasks compared to basic Python string operations?

spaCy significantly improves NLP tasks over basic Python string operations

- Efficiency and Accuracy:
    - spaCy uses pre-trained models for tokenization, part-of-speech  tagging, and named entity recognition (NER), delivering faster and more accurate results than manual string operations like regex.
- Advanced Features:
    - spaCy offers built-in capabilities like NER, dependency parsing, and word embeddings, which are complex to implement with string operations.
- Scalability:
    - spaCy efficiently processes large datasets and supports complex NLP pipelines while string operations are error-prone and impractical for scale.

2. Comparative Analysis: Scikit-learn vs. TensorFlow

| **Aspect** | **Scikit-learn** | **TensorFlow** |
|---|---|---|

| | | |
|---|---|---|
| Target Applications | Best for classical machine learning. Suited for small-to-medium tabular datasets, such as classification or clustering tasks. | Designed for deep learning and large-scale tasks like image recognition, NLP, or time-series analysis. Optimized for GPU acceleration. |
| Ease of Use for Beginners | Highly beginner-friendly with simple APIs, minimal setup, and clear documentation. Requires less mathematical knowledge. | Steeper learning curve due to complex graph-based operations. Keras API simplifies usage, but deep learning concepts are challenging for novices. |
| Community Support | Strong community with extensive tutorials, Stack Overflow support, and integration with pandas/Numpy. Less focus on cutting-edge research. | Massive community with industry and research support, active forums, and resources like TensorFlow Hub. Excels in deep learning advancements. |

---

Ethics and Optimization

1. Ethical Considerations

Potential Biases in Models:

- MNIST Handwritten Digits Model:
    - Bias could arise if the dataset lacks diversity in handwriting styles leading to poor generalization for certain groups.
    - Example: The model may struggle with non-standard digit styles from underrepresented demographics, causing unfair performance disparities.
- Amazon Product Reviews Model:
    - Bias may occur due to imbalanced sentiment data (e.g more positive reviews) or demographic skews in reviewer profiles, affecting sentiment analysis or NER accuracy.
    - Example: If reviews are predominantly from one demographic, the model may misinterpret sentiments or entities in reviews from other groups.

Mitigation with Tools:

- TensorFlow Fairness Indicators:

    - This tool evaluates model fairness by analyzing metrics (e.g., false positive rates) across demographic slices. For MNIST, it can identify performance gaps for specific handwriting styles and suggest retraining with balanced data.
    - For Amazon Reviews, it can assess sentiment prediction fairness across user groups, ensuring equitable outcomes.
- spaCy's Rule-Based Systems:
    - spaCy's rule-based NER can be customized to include diverse entity patterns, reducing bias in entity extraction for Amazon Reviews.
    - Example: Adding rules for multilingual brand names ensures better recognition across global products, mitigating cultural bias.

2. **Troubleshooting Challenge**

While implementing the Named Entity Recognition (NER) and rule-based sentiment analysis on the Amazon product reviews dataset using spaCy, several challenges and limitations were encountered:

**1. Inaccurate Entity Recognition**

- **Problem**: spaCy's pre-trained en_core_web_sm model often failed to detect product names (e.g., "AirPods", "Galaxy S21") as PRODUCT entities.
- **Reason**: The model is not specifically trained on e-commerce or product review data, so its vocabulary and entity recognition are optimized for general-purpose text, not brand or product detection.
- **Solution**: As a workaround, we accepted ORG labels as proxies for product-related entities. For more accurate results, custom training or fine-tuning the NER model on product-specific data would be needed.

**2. Sentiment Misclassification**

- **Problem**: The rule-based sentiment analysis missed nuanced expressions of sentiment. For example, sarcasm, double negatives, or mixed reviews could not be interpreted correctly.
- **Example**: A sentence like "I thought it would be great, but it turned out to be awful" might only trigger one sentiment label depending on which keyword is matched first.
- **Solution**: We used keyword-based matching with spaCy's Matcher, which is simple and works for basic polarity. However, to improve accuracy, integrating a sentiment analysis

library like **VADER** or **TextBlob**, which considers context and intensity, would be a better alternative.

**3. Dataset Quality and Cleaning**

- **Problem**: The real-world Amazon reviews dataset had missing or inconsistent entries, such as null values, non-string types, or overly long reviews.
- **Solution**: We pre-processed the data by selecting the first 100 non-null reviews and converting all inputs to strings. For a production-grade system, more advanced text preprocessing (like language filtering, de-duplication, or lemmatization) would be required.

**4. Performance Bottlenecks**

- **Problem**: Processing large volumes of reviews with spaCy's pipeline can be computationally expensive.
- **Solution**: For this assignment, only a subset (100 reviews) was processed to keep it efficient. In real applications, processing can be optimized by disabling unnecessary pipeline components or using batch processing with nlp.pipe().