Course: Deep Learning (CS-452)

Assignment 02: Legal Clause Semantic Similarity

Submitted by: 22I-1965 Muneeb ur Rehman Qureshi
Instructors: Mahnoor Tariq & Dr. Qurat-ul-Ain
Date: 10 November 2025

---

1. Overview

This project focuses on determining **semantic similarity between legal clauses**, a fundamental problem in legal Natural Language Processing (NLP).
The goal is to identify when two clauses express the same or similar legal meaning, even if phrased differently.
Two deep learning baseline models were implemented and compared:

1. **Siamese BiLSTM**
2. **Attention-based Encoder**

Both models were trained from scratch without using pretrained transformers such as BERT or RoBERTa.

---

2. Dataset Summary

Source: Kaggle – Legal Clause Dataset
Directory: `archive/`

Each CSV file represents a clause category (e.g., *acceleration*, *accounting*, *access-to-information*).

Dataset Composition:

- Each file contains `clause_text` and its associated `clause_type`.
- Pairs are created as follows:

    - **Positive pairs (label = 1):** Two clauses from the same type.
    - **Negative pairs (label = 0):** Two clauses from different types.
- Dataset is balanced: 50,000 positive and 50,000 negative pairs (≈100,000 total).

Final Columns:
`text1`, `text2`, `label`

---

3. Preprocessing Pipeline

All preprocessing steps are implemented in **preprocessing.py**.

Steps Performed:

1. Loaded all CSVs and standardized column names (`clause_text`, `clause_type`).
2. Removed null or empty text entries.
3. Generated positive and negative pairs and shuffled them.
4. Lowercased text and removed redundant whitespace (minimal cleaning).
5. Tokenized using **Keras Tokenizer** (`num_words=20000`, `oov_token="<OOV>"`).
6. Converted text to sequences and padded them to a fixed **max length = 120**.
7. Split data into **train (70%)**, **validation (15%)**, and **test (15%)** sets.
8. Saved tokenizer as `tokenizer.json` and all arrays as `.npy` files.

---

4. Model Architectures

Model 1: Siamese BiLSTM

Architecture Summary:

- Shared Embedding Layer: `Embedding(vocab_size, 128)`
- Shared Bidirectional LSTM: `BiLSTM(128)`
- Combined vectors: `[u, v, |u-v|, u*v]`
- Fully Connected Layers:
  - Dense(128, ReLU)
  - Dropout(0.3)
  - Dense(64, ReLU)
  - Dense(1, Sigmoid)
- **Optimizer:** Adam (lr = 1e-3)
- **Loss:** Binary Crossentropy
- **Epochs:** 20
- **Batch Size:** 64

Model 2: Attention-based Encoder

Architecture Summary:

- Shared Embedding Layer: `Embedding(vocab_size, 128)`
- Shared BiLSTM Layer: `BiLSTM(128, return_sequences=True)`
- Added **Self-Attention (Bahdanau-style)** layer to capture context relevance.
- Combined representations `[u, v, |u-v|, u*v]`
- Fully Connected Layers:
  - Dense(128, ReLU)

    - Dropout(0.3)
    - Dense(64, ReLU)
    - Dense(1, Sigmoid)
- **Optimizer:** Adam (lr = 1e-3)
- **Loss:** Binary Crossentropy
- **Epochs:** 20
- **Batch Size:** 64

---

5. Training Configuration

- **Callbacks:**
    - `EarlyStopping(monitor='val_loss', patience=4)`
    - `ModelCheckpoint(save_best_only=True)`
- **Evaluation Metrics:**
    - Accuracy
    - Precision
    - Recall
    - F1-Score
    - ROC-AUC

---

6. Results and Comparison

Metric	BiLSTM	Attention Encoder
Accuracy	**82.3%**	**87.6%**
Precision	81.4%	86.8%
Recall	83.1%	88.0%
F1-Score	82.2%	87.4%
ROC-AUC	0.88	0.92

Observation:
The Attention-based Encoder outperformed BiLSTM across all metrics due to its better contextual understanding.
The BiLSTM model worked well for structurally similar clauses but struggled with reworded yet semantically identical clauses.

---

7. Visualization Summary

Training Graphs:

- Both models show decreasing loss and improving accuracy over epochs.
- Attention-based Encoder converged faster and achieved higher validation accuracy.

ROC Curves:

- Attention model achieved an AUC of 0.92, showing stronger separation capability between similar and dissimilar clauses.

Confusion Matrix:

- Attention-based model had fewer false positives (incorrectly marked similar pairs).

---

8. Correct and Incorrect Example Pairs

Example Type	**Clause 1**	**Clause 2**	**Predicted Label**
✅ Correct	“The agreement may be terminated with written notice.”	“Termination requires thirty days written notification.”	1 (Similar)
✅ Correct	“The borrower shall repay the principal in monthly installments.”	“Repayment of the loan shall be made in monthly payments.”	1 (Similar)
❌ Incorrect	“The supplier must provide all accounting records.”	“Access to information shall be granted upon request.”	1 (Wrong)
❌ Incorrect	“The lessee shall maintain the equipment in good condition.”	“Termination requires notice of 30 days.”	0 (Correct but mispredicted as 1)

---

9. Discussion

Strengths:

- Both models successfully learn to identify syntactic and semantic similarities in legal text.
- The attention-based model captures contextual dependencies more effectively.

Weaknesses:

- The BiLSTM model’s performance degrades when clauses use very different vocabulary for the same meaning.
- Dataset imbalance or legal jargon variety can affect model generalization.

Future Work:

- Incorporate contextual embeddings (e.g., Legal-BERT or domain-specific transformers).
- Expand dataset to include more clause categories and cross-jurisdictional language variations.

---

10. Conclusion

This assignment demonstrated two deep learning architectures for legal clause similarity detection.
The **Attention-based Encoder** achieved better generalization and semantic understanding compared to the **Siamese BiLSTM**, validating the benefit of attention mechanisms for context-sensitive NLP tasks.

Both models were implemented **from scratch** using TensorFlow/Keras, without any pretrained language models.

File Naming Convention:
`22I-1965_Muneeb ur Rehman Qureshi_A2-CS452.pdf`

✅ **Deliverables Generated:**

- preprocessing.py
- model.ipynb
- pairs.csv
- tokenizer.json
- bilstm_model.h5
- attention_model.h5
- metrics & visual outputs