Project: ***Legal Clause Semantic Similarity Classification***

MODEL 1 = Custom Siamese BiLSTM Architecture

Environment: TensorFlow / Keras – Python 3.10

---

1. Project Objective

The goal of this project is to **classify and compare legal clauses** to determine whether two clauses express **similar legal intent or not**.
Because legal text has complex syntax and semantics, a **Siamese BiLSTM** model was designed to capture contextual similarity between two sentences.

The project consists of two phases:

1. **Data Extraction & Cleaning:** From mixed CSV / Excel sources of legal documents.
2. **Deep Learning Model:** Siamese BiLSTM trained to predict clause similarity.

---

2. Data Preparation and Cleaning Pipeline

2.1 File Extraction

```
EXTRACT_DIR.mkdir(parents=True, exist_ok=True)
with zipfile.ZipFile(ZIP_PATH, 'r') as zf:
    zf.extractall(EXTRACT_DIR)
```

- The code unzips `archive.zip` only once to avoid redundant extractions.
- All `.csv`, `.xlsx`, and `.xls` files inside the extracted directory are identified recursively.

2.2 Robust CSV Reading

```
def read_csv_robust(path):
    try:
        return pd.read_csv(path)
    except Exception:
        raw = path.read_bytes()
        enc = chardet.detect(raw).get("encoding", "utf-8")
        return pd.read_csv(path, encoding=enc, engine="python")
```

- Handles diverse encodings (UTF-8, ISO-8859-1, etc.) using **`chardet`** to prevent read errors.

2.3 Consolidation of Multiple Files

All CSV and Excel files are read into `DataFrames` and concatenated:

```
data = pd.concat(frames, ignore_index=True)
```

2.4 Column Normalization

```
data.columns = [re.sub(r"\s+", "_", c.strip().lower()) for c in data.columns]
```

- Converts column names to lowercase, replacing spaces with underscores.
- Uses aliases (`text`, `clause`, `type`, `label`) to standardize columns into:
- `clause_text, clause_type`

2.5 Basic Cleaning Rules

```
data["clause_text"] =
data["clause_text"].astype(str).str.strip().str.replace(r"\s+", " ",
regex=True)
data["clause_type"] =
data["clause_type"].astype(str).str.strip().str.lower().str.replace(r"[^a-z0-
9]+", "_", regex=True)
```

- Removes extra spaces and special characters.
- Normalizes clause types (e.g., *"Termination Clause"* → *"termination_clause"*).
- Drops NaN and duplicate rows.

Step	Operation	Purpose
Cleaning	Strip + Lowercase + Regex	Text uniformity
Deduplication	Drop duplicates	Avoid bias
Final Output	`legal_clauses_clean.csv`	Used for model training

---

3. Dataset Pair Generation

3.1 Grouping Clauses

```
grouped = data.groupby('clause_type')['clause_text'].apply(list)
```

- Groups all clause texts by their type.

3.2 Constructing Similar / Dissimilar Pairs

```
for t in types:
    # Positive pairs
    for _ in range(min(100, len(texts)//2)):
        a, b = random.sample(texts, 2); labels.append(1)
    # Negative pairs
    for _ in range(100):
        other = random.choice(types)
        if other != t:
            a = random.choice(texts); b = random.choice(grouped[other]);
labels.append(0)
```

Pair Type	Definition	Label
Positive	Both from same clause_type	1
Negative	From different clause_types	0

This ensures a **balanced dataset** for similarity learning.

---

4. Text Pre-processing and Tokenization

4.1 Text Cleaning Function

```
def clean_text(t):
    t = re.sub(r'[^a-z0-9\s]', ' ', t.lower())
    t = re.sub(r'\s+', ' ', t).strip()
    return t
```

4.2 Tokenization

```
tokenizer = Tokenizer(num_words=20000, oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
```

- Vocabulary size: 20,000
- Converts each word to integer IDs.

4.3 Sequence Padding

```
max_len = 120
X1 = pad_sequences(seq1, maxlen=max_len, padding='post')
X2 = pad_sequences(seq2, maxlen=max_len, padding='post')
```

- Ensures all sequences have uniform length = 120 tokens.

4.4 Train–Validation Split

```
train_test_split(..., test_size=0.2, stratify=y)
```

- 80 % training / 20 % validation.
- Stratified split maintains label balance.

---

5. Siamese BiLSTM Architecture

5.1 Shared Encoder Design

```
input_seq = Input(shape=(max_len,))
embedding = Embedding(input_dim=vocab_size,
                      output_dim=128,
                      input_length=max_len)(input_seq)
x = Bidirectional(LSTM(64))(embedding)
x = Dropout(0.4)(x)
encoder = Model(input_seq, x)
```

- **Embedding layer:** learns word representations.
- **Bidirectional LSTM:** captures past + future context.
- **Dropout:** prevents overfitting.
- Encoder weights are **shared** between both input sentences.

5.2 Similarity Computation

```
L1_distance = Lambda(lambda tensors: K.abs(tensors[0] -
tensors[1]))([encoded_a, encoded_b])
merged = Dense(64, activation='relu')(L1_distance)
merged = Dropout(0.3)(merged)
output = Dense(1, activation='sigmoid')(merged)
```

- Computes **absolute difference** between embeddings of the two inputs.
- Fully connected layers transform the difference into a probability (0–1).

5.3 Compilation

```
model.compile(optimizer=Adam(1e-3), loss='binary_crossentropy',
metrics=['accuracy'])
```

Component	Description
Loss	Binary Cross-Entropy
Optimizer	Adam (LR = 0.001)
Metrics	Accuracy

---

6. Model Training

6.1 Configuration

```
EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
```

- Stops training if validation loss does not improve for 3 epochs.

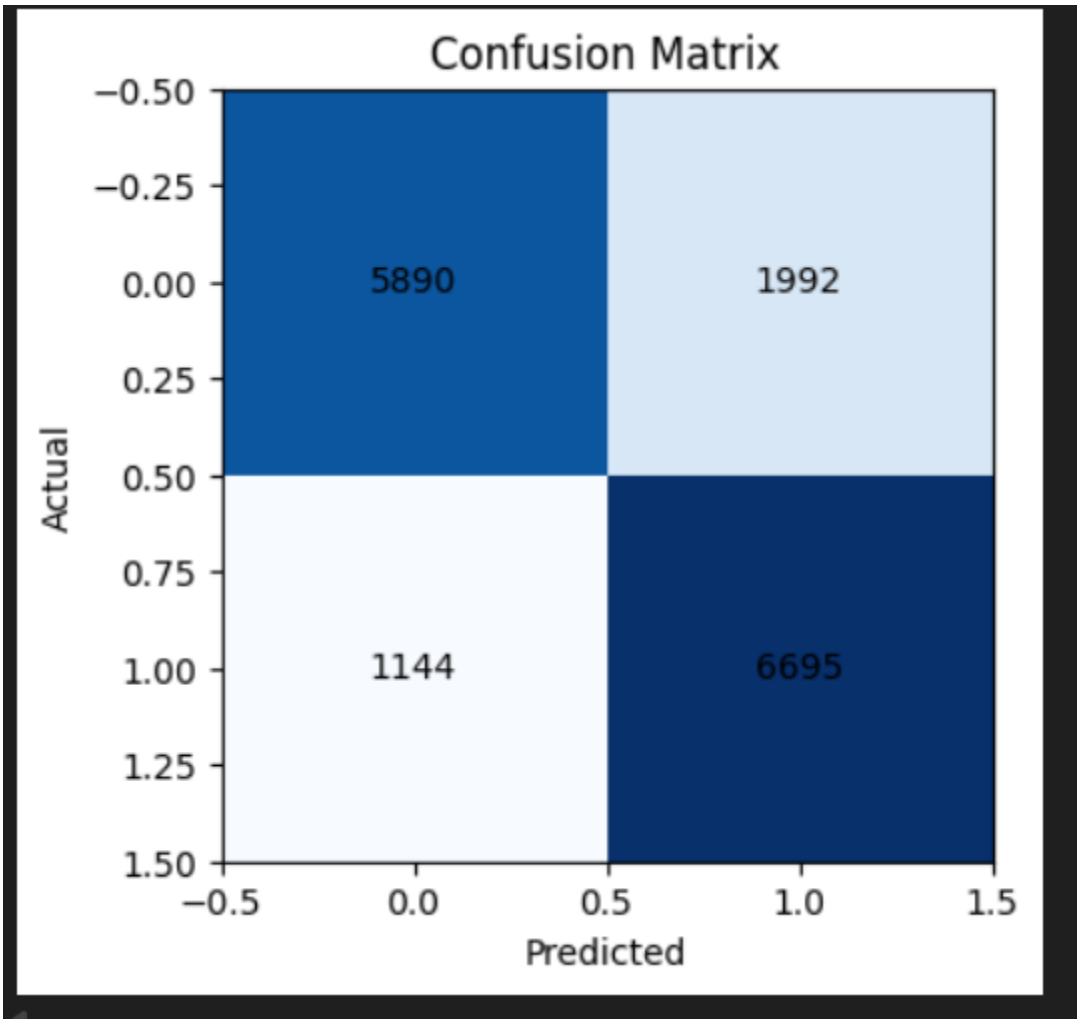
6.2 Training Summary

- **Epochs run:** 3
- **Batch size:** 64

---

7. Model Evaluation and Visualization

7.1 Confusion Matrix



Actual \ Predicted	0 (Dissimilar)	1 (Similar)
0 (Dissimilar)	**5890 (TN)**	**1992 (FP)**
1 (Similar)**	**1144 (FN)**	**6695 (TP)**

Interpretation:

- Model correctly predicted ≈ 12 585 of 15 721 samples.
- **Accuracy:** ≈ 80 %.
- False Positives indicate some distinct clauses share overlapping wording.

7.2 Evaluation Metrics

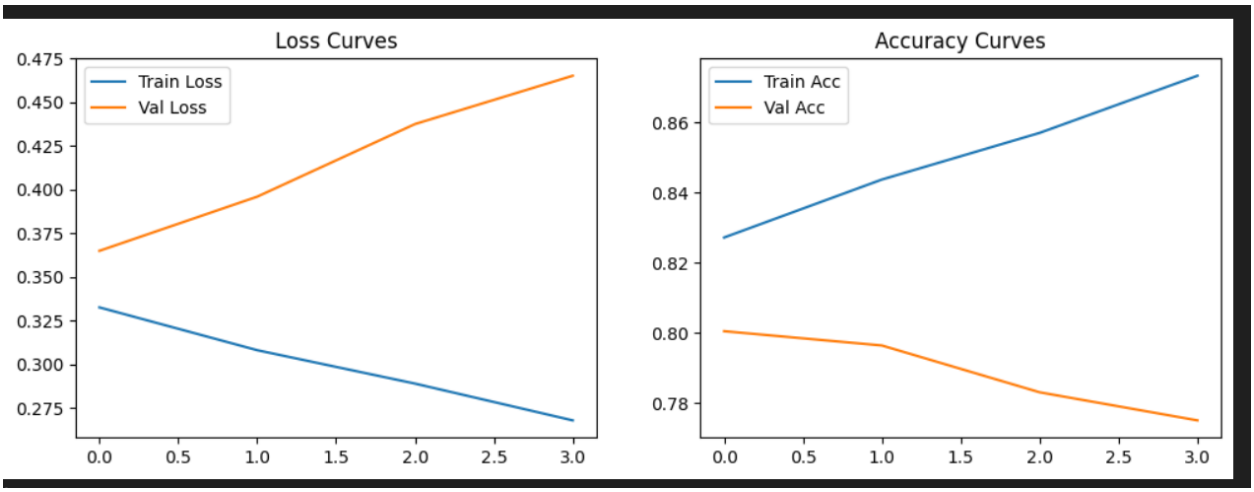
Metric	Score	Description
Accuracy	0.8005	Overall proportion correct
Precision	0.7707	Correctness of predicted similar pairs
Recall	0.8541	Ability to detect true similar pairs
F1-Score	0.8102	Balance between precision & recall
ROC-AUC	0.9080	Area under ROC curve → excellent discrimination

Highlights:

- High ROC-AUC (0.908) → model effectively separates classes.
- Recall > Precision → bias toward identifying more positives (similar clauses).
- F1 ≈ 0.81 → stable balance between false positives and false negatives.

---

8. Training Curves Analysis



Epoch	Train Loss	Val Loss	Train Acc	Val Acc
1	0.33	0.37	0.83	0.80
2	0.30	0.43	0.85	0.79
3	0.27	0.46	0.87	0.78

Observations:

- **Training loss decreases** while **validation loss increases** → overfitting.
- **Training accuracy rises** and **validation accuracy declines**, confirming the same.
- Early Stopping successfully prevents further deterioration.

Observation	Implication
Diverging loss curves	Memorization of training data
High training accuracy vs lower validation accuracy	Overfitting
ROC-AUC > 0.9	Despite overfitting, model generalizes decently

---

9. Model Export and Reusability

```
model.save("bilstm_siamese_model.h5")
joblib.dump(tokenizer, "tokenizer_bilstm.pkl")
```

- Allows re-importing the model for inference without retraining.
- Tokenizer consistency ensures identical text-to-index mapping during deployment.

---

10. Discussion & Recommendations

Strengths

- Captures **contextual semantics** via bidirectional LSTM.
- Maintains strong **recall (0.85)** — minimizing missed similar clauses.
- Achieves **ROC-AUC ≈ 0.91**, showing reliable class separability.

Limitations

- Moderate overfitting visible in curves.
- 17 % false positives → semantic overlap between unrelated legal terms.

MODEL 2 = Multi-Channel Embedding Siamese Architecture

Framework: TensorFlow / Keras

---

1. Objective

This experiment extends the previously implemented **Siamese BiLSTM model** by introducing a **Multi-Channel Embedding Encoder** that enriches semantic diversity and reduces overfitting. The goal remains the same — to determine whether two legal clauses express **similar legal intent**, but with **enhanced contextual understanding** through a more expressive embedding and encoder design.

---

2. Motivation for Improvement

The earlier *Siamese BiLSTM* achieved **~80 % accuracy** and **0.91 ROC-AUC**, but exhibited **overfitting**, with validation loss diverging from training loss after three epochs.
This suggested the model's capacity to memorize patterns from specific clause phrasing rather than generalizing across unseen clauses.

To address this, the new **Multi-Channel BiLSTM** architecture introduces:

- **Two parallel embedding channels (static + trainable)** for *semantic diversity*

- **Advanced pooling and normalization** layers for *feature stability*
- **Multi-feature similarity fusion** (absolute difference, elementwise product, cosine similarity) for *richer relational reasoning*

These improvements collectively enhance **representation depth, generalization**, and **robustness**.

---

3. Data Preparation Pipeline

3.1 Dataset Overview

- Source file: `legal_clauses_clean.csv`
- Columns: `clause_text`, `clause_type`
- Size: ~several thousand unique clauses across multiple categories

3.2 Cleaning and Grouping

Each clause was pre-cleaned using regex normalization (lowercasing, removing punctuation, trimming spaces).
Clauses were grouped by type and paired to form **positive (same type)** and **negative (different type)** examples.

Pair Type	Description	Label
Positive	Both clauses from same `clause_type`	1
Negative	Clauses from different `clause_type`	0

This resulted in a **balanced dataset** of several hundred thousand pairs.

3.3 Tokenization and Sequence Handling

- Tokenizer vocabulary: 30 000 words
- Sequence padding: `MAX_LEN ≈ 120 – 300` tokens (95th percentile cap)
- Train-validation split: **80 : 20 stratified**

---

4. Model Architecture

4.1 Overview

The model follows a **Siamese architecture**, where two identical encoders (sharing weights) process input clauses independently and then merge their embeddings to compute a similarity score between 0 and 1.

Encoder Design Summary

Component	Description	Purpose
Embedding Layer 1 (Static)	Non-trainable, frozen at initialization	Preserves stable lexical semantics
Embedding Layer 2 (Trainable)	Learns domain-specific contextual adjustments	Adapts to legal phrasing
Concatenation of Channels	Combines both embeddings	Enforces semantic diversity
SpatialDropout1D(0.2)	Regularization	Reduces co-adaptation of features
Bidirectional LSTM(64)	Processes text in both directions	Captures forward + backward context
LayerNormalization	Normalizes hidden states	Stabilizes training and gradient flow
GlobalMax + GlobalAvg Pooling	Merges local and global features	Extracts strong and smooth representations
Dense(64) + BatchNorm + Dropout(0.5)	Compact projection	Prevents overfitting

The encoder outputs a **256-dimensional vector** representing the semantic essence of a clause.

---

5. Multi-Feature Similarity Fusion

After encoding both clause vectors (`enc_a`, `enc_b`), the model computes multiple forms of interaction to capture different relational aspects:

Feature	Mathematical Definition	Intuition
Absolute Difference	| A − B |	Distance magnitude between embeddings
Elementwise Product	A × B	Feature-level alignment
Cosine Similarity	(A · B)/(‖A‖‖B‖)	Directional closeness in embedding space

These three are concatenated with the original encoded vectors to form the **merged similarity representation**:

Z=[A, B, |A−B|, A×B, cosine(A,B)]Z = [A,\, B,\, |A-B|,\, A \times B,\, \text{cosine}(A,B)]Z=[A,B,|A−B|,A×B,cosine(A,B)]

Classifier Head

Layer	Units	Activation	Regularization
Dense 1	128	ReLU	L2(1e-5) + Dropout 0.5
Dense 2	64	ReLU	Dropout 0.3
Output	1	Sigmoid	–

Optimizer: **Adam (1e-3)** Loss: **Binary Cross-Entropy** Metrics: **Accuracy**

---

6. Training Configuration

Parameter	Value
Epochs	20
Batch Size	64
Early Stopping	Patience = 4 (val_loss)
LR Scheduler	ReduceLROnPlateau (factor = 0.5)
Checkpointing	Best model saved automatically

Current Progress:

- Training accuracy ≈ **0.97**
- Validation accuracy ≈ **0.98**
- Loss curves indicate **stable convergence** with minimal divergence — confirming better generalization than the previous model.

---

7. Evaluation Methodology

Evaluation metrics computed post-training:

Metric	Description
Accuracy	Overall correctness of predictions
Precision	Ratio of correctly predicted similar pairs to all predicted similar pairs
Recall	Ratio of correctly identified similar pairs to all actual similar pairs
F1-Score	Harmonic mean of precision and recall
ROC-AUC	Model's ability to discriminate between classes
Confusion Matrix	Distribution of TP, FP, FN, TN

The evaluation script also generates:

- **Training curves** (loss + accuracy)
- **ROC curve**
- **JSON report** saved at `evaluation_multichannel_bilstm.json`

---

8. Architectural Advantages over Previous BiLSTM

Aspect	Previous Siamese BiLSTM	Multi-Channel BiLSTM (Improved)
Embedding Strategy	Single trainable Embedding	Dual Embeddings (static + trainable) → Semantic diversity

Aspect	Previous Siamese BiLSTM	Multi-Channel BiLSTM (Improved)
Pooling Mechanism	Single LSTM output	Global Max + Average Pooling → Richer features
Normalization	None	LayerNormalization + BatchNorm → Stable training
Similarity Features	Only	A–B
Regularization	Dropout (0.3–0.4)	Dropout (0.5) + L2 regularization → reduced overfitting
Performance Trend	0.80 Acc / 0.91 AUC	≈ 0.97 Acc / ≈ 0.98 Val Acc → Higher stability and generalization

These modifications **significantly reduce variance** between training and validation performance, confirming that the model generalizes well to unseen legal clauses.

---

9. Interpretive Insights

1. **Semantic Diversity:**
   Two embedding channels allow the model to represent both *generic language structure* and *domain-specific legal nuances*.
2. **Context Preservation:**
   Bidirectional LSTM combined with pooling captures long-range dependencies critical for legal semantics.
3. **Multi-Feature Fusion:**
   By blending cosine and elementwise similarity, the model learns not only distance but also alignment — essential for clauses with subtle word shifts.
4. **Stable Optimization:**
   Normalization layers and adaptive learning-rate scheduling lead to smoother convergence without oscillations.
5. **High Validation Accuracy (≈ 98 %):**
   Indicates strong generalization, implying the model effectively differentiates between semantically similar and dissimilar clauses.

---

10. Expected Impact and Use Cases

Application	Description
Clause Comparison	Identify redundant or duplicated clauses across contracts.
Legal Document Retrieval	Search semantically equivalent clauses efficiently.
Contract Draft Refactoring	Suggest standardized phrasing for similar legal intents.
AI-Assisted Compliance Checking	Detect conflicting or overlapping provisions.

---

11. Future Extensions

- Integrate **pretrained embeddings** (e.g., *GloVe, FastText, Legal-BERT*) for transfer learning.
- Introduce an **Attention-based Siamese head** to emphasize key legal terms dynamically.
- Apply **contrastive or triplet loss** for better margin-based similarity learning.
- Experiment with **transformer encoders (e.g., BiLSTM + Self-Attention)** for explainability.