<i221943> Haider Farooq — A2-CS452

1. Background

Legal documents are written in highly formal, structured language, often using complex terminology to express specific rights, duties, and conditions. However, the same legal principle can be worded in multiple ways across different laws, contracts, or jurisdictions. Legal clause similarity focuses on identifying when two clauses convey the same or closely related meanings, even if their wording differs.
This task is essential in applications such as contract analysis, case law retrieval, and legal document comparison, where determining semantic equivalence can save significant time and prevent redundancy or conflict.
Understanding clause similarity requires not only lexical matching but also deep comprehension of legal semantics, context, and logical relationships that makes it a challenging yet valuable problem in the field of legal NLP.
In the context of legal clause similarity, the underlying goal is to quantify the semantic relationship between two legal clauses in a measurable way. Instead of dealing with emotions or visual cues as in affective computing, this task involves the representation of textual meaning through embeddings and attention-based mechanisms that capture both lexical and contextual similarity.
The similarity between legal clauses can be viewed along two key dimensions:

- Semantic Equivalence: Whether two clauses express the same legal principle or rule, even if phrased differently.
- Contextual Relatedness: Whether two clauses address related topics or legal concepts, though not identical in meaning.

2. Notes for Dataset

Each CSV file name denotes a distinct clause category (such as acceleration, access-to-information, or accounting-terms). Within each file, the dataset provides multiple clause texts belonging to that category. Each clause entry includes two fields — the clause text itself and its corresponding clause type label. The overall dataset thus forms a collection of clause groups, enabling the study of semantic similarity both within and across clause categories.

3. Task & Constraints

We develop an NLP model to identify semantic similarity between legal clauses in the given dataset, under the following constraints:

- Implement at least two different baseline architectures (no pre-trained transformers or fine-tuned legal models).
- Train and evaluate with standard NLP metrics (Accuracy, F1, ROC-AUC, PR-AUC, etc.).
- Provide a comparative analysis, including strengths/weaknesses.

Baselines implemented

1. Siamese BiLSTM sentence encoder.
2. Siamese CNN + Additive Self-Attention encoder (non-transformer).

---

4. Dataset Splits

Clauses were split before pair construction to prevent leakage. Positive pairs = same clause type; Negative pairs = different types (balanced 1:1).

- Clauses — Train: 105,612 | Val: 22,636 | Test: 22,633
- Pairs — Train: 211,224 | Val: 45,270 | Test: 45,266

This balancing justifies reporting Accuracy alongside F1 and AUC metrics.

---

5. Network Details (Architectures, Parameters, Training Settings)

Common

- Tokenizer/Vocab: Keras Tokenizer, num_words=30,000, <OOV> token
- Max length: 308 (95th percentile heuristic on train)
- Embeddings: Random initialized, 128-d, mask_zero=True

- Siamese head (both models): concat[|u−v|, u*v, u, v] → Dense(256, ReLU) → Dropout(0.2) → Dense(64, ReLU) → Dense(1, Sigmoid)
- Loss & Metrics: Binary Cross-Entropy; Accuracy, Precision, Recall, ROC-AUC, PR-AUC
- Optimizer/LR: Adam, lr=2e-3, ReduceLROnPlateau; EarlyStopping on val ROC-AUC
- Batch size / Epochs: 128 / up to 8 (early stop triggered)
- Rationale: Siamese setup learns pairwise similarity; no external pre-training, staying within constraints.

Baseline A — BiLSTM Encoder

- Encoder: Embedding → BiLSTM(128, return_sequences=True) → GlobalMaxPool1D → Dense(128, ReLU)
- Strengths: Captures sequential dependencies & long-range context; robust to paraphrase structure.

Baseline B — CNN + Self-Attention Encoder (Non-Transformer)

- Encoder: Embedding → Conv1D(128, k=3/4/5) (ReLU) → Concat → Additive Self-Attention (mask-aware) → Dense(128, ReLU)
- Strengths: Efficient n-gram pattern extraction; attention highlights salient tokens; typically faster per epoch.

---

6. Training Graphs

(Insert the four figures from your notebook output here.)

- Figure 1. BiLSTM — Training vs Validation Loss
- Figure 2. BiLSTM — Training vs Validation Accuracy
- Figure 3. CNN+Self-Attention — Training vs Validation Loss
- Figure 4. CNN+Self-Attention — Training vs Validation Accuracy

Observation: Both models converge rapidly with near-ceiling validation accuracy; BiLSTM shows slightly smoother validation behavior after LR reduction.

---

7. Performance Measures (Test Set)

7.1 Metric Definitions (per rubric)

- Accuracy: Overall correctness (appropriate here due to 1:1 balanced pairs).
- Precision: Fraction of predicted "similar" pairs that are truly similar (controls false positives).
- Recall: Fraction of truly similar pairs retrieved (controls false negatives).
- F1-Score: Harmonic mean of Precision and Recall (balance).
- ROC-AUC: Threshold-free separability across ROC curve.
- PR-AUC: Precision-Recall area; especially informative when positives are rare in deployment.

7.2 Results (exact values from your runs)

Metric	BiLSTM (Test)	CNN + Self-Attn (Test)
Accuracy	0.9994	0.9983
Precision	0.9990	0.9984
Recall	0.9997	0.9982
F1-Score	0.9994	0.9983
ROC-AUC	0.9999	0.9996
PR-AUC	0.9998	0.9997

(Per-class classification reports were also produced in the notebook.)

---

8. Performance Comparison of NLP Architectures (Accuracy & Time)

Model	Accuracy (Test)	Approx. Train Time / Epoch (Colab GPU)
BiLSTM	0.9994	~134–138 s
CNN + Self-Attn	0.9983	~88–110 s

Takeaways:

- BiLSTM leads slightly in accuracy/F1/ROC-AUC/PR-AUC.
- CNN+Self-Attn trains ~25–35% faster/epoch, with competitive metrics.
- In latency-sensitive scenarios, CNN+Attn is attractive; for maximal quality, BiLSTM edges out.

9. Discussion on Evaluation Metrics (Domain Rationale)

In real-world contract repositories, truly similar clauses are often sparse relative to all possible pairs. While our balanced test set legitimizes Accuracy and F1, a production system should emphasize PR-AUC and operate at a high-precision threshold to minimize costly false matches (erroneously conflating distinct legal clauses). ROC-AUC remains useful to gauge separability, but PR-AUC better reflects performance under class imbalance. Thresholds should be calibrated to legal risk tolerance (e.g., favor high precision for clause deduplication; favor higher recall in precedent retrieval).

10. Qualitative Results (Correct & Incorrect Matches)

a) Correct Positives (Similar)

1. “Time is of the essence …” ↔ “… time shall be of the essence …” (near-paraphrase of legal principle).
2. “Now, therefore in consideration … agree as follows:” ↔ “Now, therefore … intending to be legally bound … agree as follows:”

b) Correct Negatives (Different)

1. “Distributions … dividends or repurchases …” ↔ “Waivers … will not be considered waived unless …” (distinct topics).

2. “Survival … provisions shall survive …” ↔ “Severance … unenforceable provision to be severed …”

c) False Positives (Predicted Similar, Actually Different)

1. “Parties … binding upon and inure to the benefit …” ↔ “Parties in interest … persons making/assisting the claim …” (lexical overlap, different legal function).
2. “Miscellaneous … certificate of authorized persons …” ↔ “Miscellaneous provisions … severability clause …” (section headers overlap; semantics differ).

d) False Negatives (Predicted Different, Actually Similar)

1. “Definition. (Rent) … monetary obligations of tenant …” ↔ “Definition. Taxes means … charges, fees, levies …” (both definitional style; different terms—model cautious).
2. “Security interest … pledgor delivers, pledges and assigns …” ↔ “Security interest … borrower grants continuing security interest …” (high conceptual similarity but varied phrasing).

---

11. Conclusion

Both baselines achieve near-ceiling performance on this dataset with a simple, from-scratch Siamese setup.

- BiLSTM slightly outperforms on ranking and F1; better for nuanced paraphrases.
- CNN + Self-Attention is faster and competitive; preferable when training/inference speed matters. For deployment, set a high-precision operating threshold and consider ensembling (e.g., probability averaging) for robustness across legal domains.

References

- keras-idiomatic-programmer (GitHub): https://github.com/GoogleCloudPlatform/keras-idiomatic-programmer

- Handbooks: https://github.com/GoogleCloudPlatform/keras-idiomatic-programmer/tree/master/handbooks

- Dataset: https://www.kaggle.com/datasets/bahushruth/legalclausedataset

End of Report