NUCES FAST, Islamabad

Deep Learning

Assignment No.02

Instructors

Mahnoor Tariq & Dr. Qurat Ul Ain

October 26, 2025

Announcement

This is to announce that your assignment on NL[] models is now available. Please make sure to complete the task and follow all given instructions. Submit your assignment before the due date.

Due Date: 10th November, 2025

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

In the given dataset of legal clauses, the following information is provided:

- 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.

Evaluation Metrics

Report your results in following evaluation metrics

For categorical classification

Name	Metric	Description
Accuracy	Accuracy	Measures how often the model correctly classifies clause pairs as "similar" or "different." Use only if dataset is balanced (roughly equal similar/not-similar pairs).
Precision	Precision	Out of all predicted similar clause pairs, how many truly convey the same legal meaning? Important when false positives (wrongly marking clauses as same) are costly.
Recall	Recall	Out of all truly similar clauses, how many did the model successfully identify? Important if missing a truly similar clause is bad (e.g., missed precedent).
F1-Score	F1-Score	Harmonic mean of Precision and Recall — balances both. Standard metric for NLP classification tasks.
ROC-AUC / PR-AUC	AUC / AUC-PR	Evaluates classifier's ability to separate similar vs non-similar pairs across thresholds. Use to measure ranking ability of clause similarity scores.

3 Task

Develop an NLP model capable of identifying semantic similarity between legal clauses in the given dataset.

You are required to:

- Design and implement at least two different baseline architectures (e.g., BiLSTM, Attention-based Encoder, or any custom architecture of your choice).
- Train and evaluate your models without using any pre-trained transformer or fine-tuned legal model.
- Compare their performance using appropriate NLP evaluation metrics (e.g., Accuracy, F1-Score, ROC-AUC, etc.).
- Provide a comparative analysis of results and discuss the strengths and weaknesses of each baseline model in handling legal clause semantics.

Deliverables

Implement your task using IDE, Python notebook or Google Colab. You may download PyCharm.

1. The code files (.ipynb) to be uploaded on your GitHub repository and the link to be uploaded on GCR. Do not share links of Google Colaboratory, rather upload on GitHub as Python Notebook.

   - The Python Notebook should also demonstrate:
     - Neatly written, documented and modular code

   Strongly encouraged to use Best Practices and Modular implementation of Image classification pipeline.

   Links (wrapped automatically):

   - `https://github.com/GoogleCloudPlatform/keras-idiomatic-programmer`
   - `https://github.com/GoogleCloudPlatform/keras-idiomatic-programmer/tree/master/handbooks`
   - keras-idiomatic-programmer (GitHub)

2. Short Report: Should contain the following:

- Network details: architecture, parameters, training setting, etc.
- Dataset splits
- Training graphs
- Performance measures
- Performance comparison of NLP architectures (accuracy & time)

Naming Convention: `<your FastID>_<your name>_A2-CS452.pdf`

Grading Rubric

The breakdown is given below.

Short Report: 50 points. Breakdown given below

- Network details which include architecture, parameters, training setting, batch size etc. A note on rationale of choosing the baseline. **[10pt]**
- If Multiple baselines are chosen, a comparison on performance of these baselines. **[10 pt]**
- Training graphs, (showing decreasing loss and increasing accuracy with number of epochs) **[10 pt]**
- Performance measures and a discussion on domain evaluation metrics — what is the rationale of using each one and in the present scenario, which should be most suited if we want to design a system which should work in the wild. **[15 pt]**
- A few of the correctly and incorrectly matched legal clauses. **[4 pt]**
- Use naming convention while submitting the PDF file **[1 pt]**

Code: 50 points. Breakdown given below.

- Code is properly **documented, modular and understandable [10 points]**
- Modular implementation of pipeline. Use functions and object-oriented implementation, as illustrated in keras-idiomatic-programmer guide. If you are using Pytorch, object-oriented implementation counts for this. **[10 points]**
- Use of appropriate data cleaning and preprocessing, on the fly or off-line **[6 points]**
- Use of multiple NLP architectures (use at least two) and appropriately shown comparison of Quantitative performance measures **[20 points]**

  (If quantitative performance measures are too low, then points will be deduced accordingly).
- Qualitative results i.e. a few of the correctly and incorrectly matched legal clauses **[4 points]**

Dataset availability link:

https://www.kaggle.com/datasets/bahushruth/legalclausedataset?select=acceleration.csv