Molecular Substructure Prediction using NMR Spectroscopy with LSTM

Kolukula Sam Navdeep

24B1303

1 Problem Definition

Nuclear Magnetic Resonance (NMR) spectroscopy is one of the most widely used analytical techniques for determining molecular structure. However, interpreting NMR spectra and identifying molecular substructures requires significant domain expertise and manual effort.

The goal of this project is to automate the prediction of molecular substructures from NMR spectroscopy data using deep learning. The problem is formulated as a supervised learning task where sequential NMR spectral data is mapped to a binary label indicating the presence or absence of a particular molecular substructure. Since NMR spectra are inherently ordered and sequential, a Long Short-Term Memory (LSTM) neural network is chosen to effectively capture long-range dependencies and contextual relationships between spectral peaks.

2 Dataset Description

Source: The dataset is provided in serialized pickle format containing experimental NMR spectroscopy results. The dataset consists of approximately 60,000 molecules.

Features:

- Carbon (^{13}C) NMR spectroscopy data
- Solvent information
- Temperature at which NMR was recorded
- Molecular representations (molecules)

Labels: Binary labels indicating the presence or absence of specific molecular substructures.

3 Data Science Pipeline

3.1 Data Cleaning

Data cleaning was a crucial step in ensuring reliable learning. All rows containing null or missing values in the carbon NMR spectrum were removed. Only the ^{13}C NMR spectrum was retained for model training, while other incomplete spectral entries were discarded. This ensured consistency and reduced noise in the input data.

3.2 Feature Engineering

No manual feature extraction was performed. The cleaned carbon NMR spectra were directly treated as sequential numerical inputs. Solvent and temperature information were implicitly encoded within the dataset representation and aligned with the spectral data.

3.3 Data Transformation and Scaling

The processed NMR spectra were converted into PyTorch tensors and reshaped into the LSTM-compatible format:

$$(\text{batch size}, \text{sequence length}, \text{number of features})$$

If the input spectrum was one-dimensional, an additional feature dimension was added programmatically.

3.4 Data Splitting

The dataset was split into training and testing subsets prior to training. The training set was used to optimize model parameters, while the test set was used exclusively for evaluating generalization performance.

3.5 Label Creation

Each molecule was assigned a binary label:

$$1 \rightarrow \text{Substructure Present}, \quad 0 \rightarrow \text{Substructure Absent}$$

4 Model Details

4.1 LSTM Architecture Explanation

The LSTM network is designed to model sequential dependencies in NMR spectroscopy data. The architecture used in this project is defined as follows:

- **Input size:** Number of features per timestep (derived from NMR spectrum dimensionality)
- **LSTM layer:** One LSTM layer with 64 hidden units
- **Hidden state size:** 64
- **Number of layers:** 1
- **Fully connected layer:** Maps the final LSTM hidden state to a single output neuron
- **Activation function:** Sigmoid activation for binary classification

The LSTM processes the NMR spectrum timestep-by-timestep and retains important spectral information in its hidden state. The output corresponding to the final timestep is passed through a fully connected layer to predict the probability of substructure presence.

4.2 Hyperparameters

- Hidden units: 64
- Number of LSTM layers: 1
- Learning rate: 1×10^{-4}
- Optimizer: Adam
- Loss function: Binary Cross Entropy
- Batch size: 32
- Epochs: 20

These hyperparameters provide a balance between learning capacity and computational efficiency.

5 Pipeline Overview

NMR Spectra → Tensor Conversion → LSTM Sequence Modeling → Fully Connected Layer → Substructure Prediction

6 Evaluation Metrics

6.1 Used Metric

Accuracy is used as the primary evaluation metric. It is defined as the ratio of correctly predicted samples to the total number of samples and is mathematically expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives.

Accuracy is well-suited for this task because the dataset is large and reasonably balanced across substructure labels. Under such conditions, accuracy provides a clear and interpretable measure of overall model performance, reflecting the model's ability to correctly identify both the presence and absence of molecular substructures.

6.2 Other Possible Metrics

To gain deeper insight into model performance, especially in cases of class imbalance or asymmetric error costs, additional metrics could be used:

- **Precision:** Measures the proportion of correctly predicted positive samples among all predicted positives.
- **Recall:** Measures the proportion of correctly predicted positive samples among all actual positives.

- **F1-score:** The harmonic mean of precision and recall, useful when balancing false positives and false negatives is critical.
- **ROC-AUC:** Evaluates the model's ability to distinguish between classes across different decision thresholds.

Although these metrics provide valuable insights, accuracy is preferred in this work due to its simplicity, interpretability, and suitability for large-scale, balanced multi-label datasets where the objective is overall correct substructure prediction rather than optimizing for a specific type of error.

7 Results and Observations

7.1 Predicted vs Actual Substructures

For a representative test molecule, the LSTM model produced the following output:

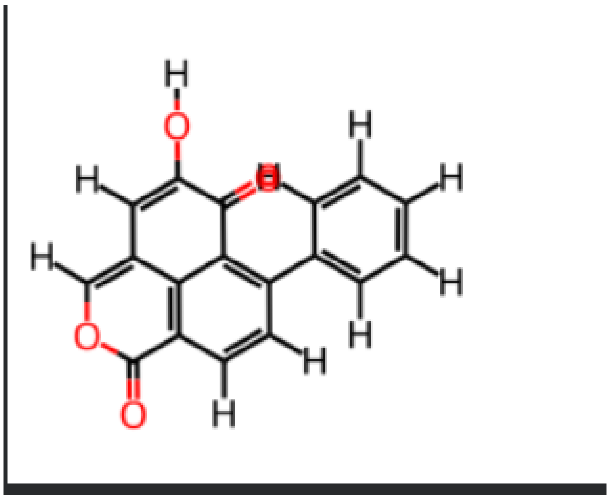
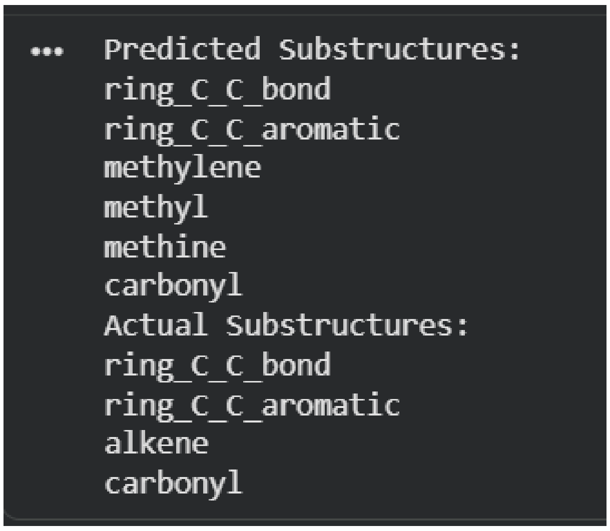


Figure 1: A test molucule



```
•••  Predicted Substructures:
     ring_C_C_bond
     ring_C_C_aromatic
     methylene
     methyl
     methine
     carbonyl
     Actual Substructures:
     ring_C_C_bond
     ring_C_C_aromatic
     alkene
     carbonyl
```

Figure 2: pred vs actual substructures

Predicted Substructures:

- ring_C_C_bond
- ring_C_C_aromatic
- methylene
- methyl
- methine
- carbonyl

Actual Substructures:

- ring_C_C_bond
- ring_C_C_aromatic
- alkene
- carbonyl

The results show that the model correctly identifies several dominant and chemically significant substructures, such as aromatic ring carbon–carbon bonds and the carbonyl group. These features are strongly represented in ^{13}C NMR spectra and are consistently learned by the LSTM across multiple samples.

At the same time, the model predicts additional aliphatic substructures such as methylene, methyl, and methine groups, while missing the alkene group present in the ground truth.

7.2 Observations

- Core structural motifs with strong and distinct NMR signatures are predicted reliably.
- The model tends to predict chemically related substructures together, especially within similar chemical shift regions.
- Predictions indicate that the LSTM learns general carbon environment patterns rather than enforcing strict chemical exclusivity.

7.3 Model Limitation

The observed mismatch between predicted and actual substructures arises primarily from the formulation of the learning objective rather than from an implementation issue. The model is trained using Binary Cross-Entropy (BCE) loss, which treats each substructure label independently. As a result, the network optimizes each label individually and does not consider the set of substructures as a chemically constrained whole.

Overall, the results demonstrate that the LSTM effectively captures dominant spectral patterns, while the minor discrepancies observed are consistent with the independent-label assumption imposed by the BCE loss formulation.

8 Code Explanation

The code is implemented using PyTorch and follows a standard deep learning workflow. First, essential libraries are imported for neural network construction, optimization, data loading, and dataset handling. The dataset is loaded from a pickle file and converted into PyTorch tensors. Since LSTM models require sequential input, the data is reshaped into three dimensions representing batch size, sequence length, and features.

A custom LSTM model class is defined using `nn.Module`. It consists of a single LSTM layer with 64 hidden units to capture sequential patterns in the ^{13}C NMR spectra. The output from the final timestep of the LSTM is passed through a fully connected layer followed by a sigmoid activation function to produce a probability score for substructure presence.

Binary Cross-Entropy loss is used because each substructure label is treated independently. The Adam optimizer is employed for stable and efficient training. The model is trained for multiple epochs using mini-batches, and accuracy is computed after each epoch. Finally, the trained model is evaluated on the test dataset with gradient computation disabled.

9 Complete Code

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import pickle

device = torch.device("cuda" if torch.cuda.is_available() else "
   cpu")

with open("dataset.pkl", "rb") as f:
    data = pickle.load(f)

x_train = torch.tensor(data["x_train"], dtype=torch.float32)
x_test  = torch.tensor(data["x_test"], dtype=torch.float32)
y_train = torch.tensor(data["y_train"], dtype=torch.float32)
y_test  = torch.tensor(data["y_test"], dtype=torch.float32)

if len(x_train.shape) == 2:
    x_train = x_train.unsqueeze(-1)
    x_test = x_test.unsqueeze(-1)

train_loader = DataLoader(
    TensorDataset(x_train, y_train),
    batch_size=32,
    shuffle=True
)

test_loader = DataLoader(
    TensorDataset(x_test, y_test),
    batch_size=32,
    shuffle=False
)

class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size=64):
        super().__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=
           True)
        self.fc = nn.Linear(hidden_size, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        out, _ = self.lstm(x)
        out = out[:, -1, :]
        out = self.sigmoid(self.fc(out))
        return out

model = LSTMModel(x_train.shape[2]).to(device)

criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=1e-4)

for epoch in range(20):
    model.train()
    correct = 0
    total = 0
    for x, y in train_loader:
        x, y = x.to(device), y.to(device).unsqueeze(1)
        optimizer.zero_grad()
        output = model(x)
        loss = criterion(output, y)
        loss.backward()
        optimizer.step()
        preds = (output > 0.5).float()
        correct += (preds == y).sum().item()
        total += y.size(0)
    print(f"Epoch {epoch+1}, Accuracy: {correct/total:.4f}")
```

10 Conclusion

This work demonstrates that LSTM networks are effective for predicting molecular substructures from NMR spectroscopy data. By modeling NMR spectra as sequential inputs, the LSTM captures both local and long-range dependencies between spectral features. The results indicate that deep learning can significantly reduce manual effort in spectral interpretation and support automated chemical analysis.

11 Data and Code Availability

- Github repo:
  https://github.com/SamNavdeep/Molecular-substructure-prediction.git
- Dataset:
  https://drive.google.com/file/d/1jIia0kHCUD7fr3VI6UFkRViSGBpGYmVg/view?usp=drive_lin