Seasons Of Code 2025

Topic- Computational Linguistics
Mentors- Sujal Sahoo, Harsh Chavda

Name- Rohan Shukla
Roll no.- 24B1274

Github repo for SOC- SOC_2025

The Learnings

This project was an invaluable experience for me. This was my first exposure to computer linguistics. Here are the things I learnt along the way.

1) Basics of NLP-
    a) This was the most energetic phase of the learning. It included following a very informative book named “NLP in Action” along with the additional learning resources provided by the mentors. The key topics learned during this phase are listed below:
        i) Overall introduction to NLP and its applications.
        ii) Pattern matching through RegEx
        iii) Tokenization
        iv) Stemming and Lemmatization
        v) Stop words

    Reference- NLP IN ACTION practice codes
2) Building upon the introduction and getting introduced to simple concepts related to the implementation of NLP.
    a) This phase was information-loaded, where we balanced multiple resources to get hands-on experience of the implementation of NLP methods.
        i) Learning the Pytorch implementation of basic NLP mechanisms like Sequential models operating on gradient descent, using different types of layers such as Linear, RNN, Dropout, Convolutional and introduction to various other tools in Pytorch such as optimizers, loss functions and activation functions.
        ii) This phase also involved building standard applications based on the learnings, such as the MNIST and CIFAR-10 classification problems.

    Reference- Basic Projects in Pytorch
3) Introduction to Transformers
    a) Here, we learned about the breakthroughs the transformer was able to achieve.
        i) We got introduced to the basics of a transformer.
        ii) We implemented a Self-attention block and tried various tweaks here and there to tinker with its basic working.
        iii) We also explored in detail the variable-length input layers, such as RNN and LSTM and implemented basic projects to assimilate them. This includes the basic Emoji prediction model.

    References- 1) Vanilla RNN implementation- Vanilla RNN implementation
    2) Emoji Prediction model- Emoji prediction

4) Mid-term and Final projects
    a) These projects were quite engaging, to be honest. Quite overwhelming at first sight for a beginner fresh into the business. The support of mentors stood out here as they helped understand the situation better and provided a starting point to delve into it.
    b) But these projects helped us understand the links between the various topics we had learnt throughout the course. The main focus was sentiment analysis through RNN and LSTM layers, while also using the additional concepts we had learnt.

    Reference- Final implementation

The Application

The final project involved analyzing the Financial Statements to classify the statements into positive, negative or neutral sentiments. The technical details are already mentioned in the Colab notebook in which the project was implemented. Here, we will focus more on how I arrived at the final project.

My first draft of the project was a messy implementation of the LSTM classifier, which was facing severe overfitting and could not generalize well. That project had several reasons to fail; however, I will mention only the issues which I was able to solve in my second and final iteration.

1) Seeing the small size of the dataset available, I thought that a full-fledged self-attention mechanism might not be trainable, and hence went with the lighter but more ambiguous alternative of sliding window mean. When trying for various parameters and still getting only 60% accuracy in unseen data, I reversed my decision in the final project to include a self-attention block.
2) I used padding to make the sentences the same length. Later, I realised that the mean length of the sentences was much lower than the maximum length of the dataset, and as a result, my training set was heavily dominated by Padding itself. To mitigate this, I accepted an unorthodox trade-off of not training by model on a GPU, which reduces the training speed considerably in return for the ability of LSTM to handle variable length sequences when passed in a batch size of 1. The small size of the dataset made this choice easier.
3) While AI helped me realise that even this small dataset was highly imbalanced with the order of samples as neutral>positive>>negative, the idea to track the F1 scores instead of the conventional equality count accuracy came to me while I was on my second attempt.
4) The second attempt also included a custom DataLoader, implemented with the help of AI to handle shuffling and splitting of variable-length datasets.
5) And of course, the second attempt was much more professional, modular and explicitly explained and compared to the first one's mess.
6) But I also acknowledge the difficulty of handling data types and branching of inputs in my second attempt, due to my choice of avoiding Padding. However, I find the improved structure and performance to be worth the pain.

References to my two tries:

1) Failed first attempt- (raw and unexplained) - Raw implementation

2) Final project- Final Project