CS563 - NLP ASSIGNMENT-2

Named Entity Recognition (NER-tagging) using HMM and RNN

Jenish Monpara 1901CS28
Tanishq Malu 1901CS63
Tarusi Mittal 1901CS65

Question: The assignment targets to implement Hidden Markov Model (HMM) to perform Named Entity Recognition (NER) task

Tags: ['B','I','O']

Tweet dataset:
https://www.dropbox.com/sh/a3q71mpth7mfoid/AABn6REap3wn7pzri-xblOOTa?dl=0

Question 1: Perform 5 fold cross-validation on the Training datasets and report both average & individual fold results (Accuracy, Precision, Recall and F-Score).

Score-Type		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Accuracy		68.63%	69.40%	68.00%	72.30%	66.92%	**69.04%**
Precision	B	0.1959	0.3107	0.2059	0.2391	0.2165	**0.2336**
	I	0.0595	0.0864	0.0669	0.0522	0.0526	**0.0635**
	O	0.9835	0.9774	0.9849	0.9824	0.9852	**0.9827**
Recall	B	0.1727	0.2388	0.1875	0.1964	0.1875	**0.1966**
	I	0.8493	0.8318	0.8372	0.7966	0.9206	**0.8471**
	O	0.6998	0.7082	0.692	0.7395	0.6807	**0.7040**
F1 Score	B	0.1836	0.27	0.1963	0.2157	0.201	**0.2133**
	I	0.1112	0.1566	0.1239	0.0979	0.0995	**0.1178**
	O	0.8177	0.8213	0.8128	0.8438	0.8051	**0.8201**

Question 2: Briefly discuss about Unigram vs Bigram assumption while training HMMs

Answer. Assumptions about the order of observed symbols can be made when training Hidden Markov Models.The unigram assumption holds that each symbol is independent of all others, whereas the bigram assumption holds that each symbol is only dependent on the preceding symbol. In practise, the bigram assumption is frequently used in HMMs because it represents many natural language and speech patterns more accurately. The unigram assumption, however, can be helpful in some circumstances where the order of symbols is less dependent on earlier symbols. Both assumptions offer distinct methods for calculating the likelihood of observing a sequence of symbols. In the end, the assumption chosen is determined by the specific context and nature of the problem being solved.

Question 2) Explain and draw the architecture of RNN that you are proposing with justification. Describe the features of RNN

Answer. For named entity recognition, we will employ bi-directional RNN.
The following are the components of the proposed architecture:
Each word in the input text is converted into a dense vector representation by an embedding layer.
A word's context in a sentence includes words that occur before and after it. A bidirectional RNN processes the input text in both forward and backward directions, enabling the model to capture contextual information from both preceding and following words.
A fully connected output layer that uses a softmax activation function to map the RNN's final hidden state to a probability distribution over entity labels.
a loss function, like cross-entropy loss, that gauges the difference between the predicted entity labels and the actual labels.

Some features of RNN are as follows:
Recurrent connections: RNNs include recurrent connections that allow information to be passed from one time step to the next. This allows the network to capture temporal dependencies in the input sequence.

Hidden state:RNNs have a hidden state that compiles data from earlier time steps and is used to forecast what will happen in the upcoming time step. Based

on the input and the previous hidden state, the hidden state is updated at each time step.

Variable-length inputs: Since the same set of weights is used for each time step, RNNs can handle variable-length inputs. They are therefore well suited for handling sequences of various lengths.

Backpropagation through time (BPTT) is a technique for training RNNs that entails unrolling the network over time and utilising the conventional backpropagation algorithm. This enables the network to develop the ability to predict outcomes based on input sequences.

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit): Memory cells can be incorporated into the RNN's architecture, as in LSTM and GRU networks that use long short-term memory. As a result, the network can choose what data from earlier time steps to keep and forget, potentially preventing the vanishing gradient problem and improve performance on long sequences.