Chua Qi Bao 1004494
Ng Peng Yu 1004269
Yap Swee En 1004340

Machine Learning Report

Part 1

Design Approach

Step 1
Find the max emission parameter for each word in the training set. The predicted label for each word will be the label that gives the highest emission parameter (since each word can be emitted by multiple labels).

$$Emission\ parameter\ = \frac{No.\ of\ times\ a\ label\ emits\ a\ word}{No.\ of\ times\ the\ label\ appears\ in\ the\ sequence}$$

Step 2
Find the max emission parameter for the #UNK# token, which is used for words in the test set which do not appear in the training set. The predicted label for #UNK# is the label that returns the highest emission parameter, among all the 7 possible labels, given k = 1.

Step 3
For every input word in the test file, we will generate a predicted label using the trained model. Then, write the output to the output file.

Results

```
chuaqibao@Chuas-MacBook-Pro Project % python3 EvalScript/evalResult.py ES/dev.out ES/dev.p1.out

#Entity in gold data: 255
#Entity in prediction: 1733

#Correct Entity : 205
Entity  precision: 0.1183
Entity  recall: 0.8039
Entity  F: 0.2062

#Correct Sentiment : 113
Sentiment  precision: 0.0652
Sentiment  recall: 0.4431
Sentiment  F: 0.1137
chuaqibao@Chuas-MacBook-Pro Project % python3 EvalScript/evalResult.py RU/dev.out RU/dev.p1.out

#Entity in gold data: 461
#Entity in prediction: 2089

#Correct Entity : 335
Entity  precision: 0.1604
Entity  recall: 0.7267
Entity  F: 0.2627

#Correct Sentiment : 136
Sentiment  precision: 0.0651
Sentiment  recall: 0.2950
Sentiment  F: 0.1067
```

Part 2

Design Approach

Step 1:
Initialise 3 dictionaries :

I. u_lab, which is number of times the specific label appears,
II. uv_lab, which the number of times the specific label transits to the next, and
III. z, the score of the of the transition

We then assign the current state to be the start. Ensure at the start of the line, we check for the start and stop state, ensuring that the code stops at the the end of the dataset, which is our consideration for q(STOP|yn) and q(y1|START). The code then reads data file line by line, taking the label by splitting the line, and then taking the last token of that line, which is the label.This is done such that the special Russian indents do not affect the reading of the data, as our initial idea was to just read the 2nd word that appears in the dataset, which would have just been the label itself. Finally Find the number of times U transitions to V, divided by the number of U appears, which is the number of times the previous label produces the next label.

$$Transition\ parameter\ =\ \frac{No.\ of\ times\ U\ goes\ to\ V}{No.\ of\ times\ the\ U\ appears}$$

Step 2:
In order to manage with underflow of numbers, we utilise Log so that our calculations are done with Sum (+) instead of multiplication (*). This ensures that the numbers grow bigger via summation, instead of growing smaller via multiplication of numbers smaller than 1, as shown below

$$Probability\ P1\ =\ 0.01$$
$$Probability\ P2\ =\ 0.05$$
$$Summation\ =\ P1\ +\ P2\ =\ 0.01\ +\ 0.05\ =\ 0.06$$
$$Log\ Summation\ =\ Log(P1)\ +\ Log(P2)\ =\ -3.30$$
$$Multiplication\ =\ P1\ *\ P2\ =\ 0.005$$

As seen, if we utilise multiplication, the numbers will grow smaller, approaching 0. However, if we utilise log summation, the numbers will still be a tangible whole number (-3.30) that is useful for calculations.

Following the pseudocode from the lecture, we obtain the highest score for each transition parameter and emission parameter that we have accomplished in Part 1 and Part 2 and train the data using them, applying it into the viterbi algorithm. After that we implement a backtrack to track the path taken, and write it into dev.p2.out.

Results

For ES

```
chuaqibao@Chuas-MacBook-Pro Project % python3 EvalScript/evalResult.py ES/dev.out ES/dev.p2.out

#Entity in gold data: 255
#Entity in prediction: 313

#Correct Entity : 163
Entity  precision: 0.5208
Entity  recall: 0.6392
Entity  F: 0.5739

#Correct Sentiment : 114
Sentiment  precision: 0.3642
Sentiment  recall: 0.4471
Sentiment  F: 0.4014
```

For RU

```
chuaqibao@Chuas-MacBook-Pro Project % python3 EvalScript/evalResult.py RU/dev.out RU/dev.p2.out

#Entity in gold data: 461
#Entity in prediction: 420

#Correct Entity : 240
Entity  precision: 0.5714
Entity  recall: 0.5206
Entity  F: 0.5448

#Correct Sentiment : 148
Sentiment  precision: 0.3524
Sentiment  recall: 0.3210
Sentiment  F: 0.3360
```

Part 3

Design Approach

Using what we have in Part 2, we added “x” to the inputs for viterbiAlgo. For every value in maxprobe list, we run through it, and check if it's higher than the current max. If the value checked is higher than the current max, the current max is removed from the list and this new value is appended into the list and becomes the next max score to be compared with future scores down the line of code. At the end of the 1st iteration, the value is stored in the bestX list and removed from masterlist maxprobe. Repeat this X (5) times, and the bestX list should have the top 5 scores. We then write the fifth best output to the file.

Results

```
(base) Swees-MacBook-Pro-2:~ ouryuuzeno$ python3 /Users/ouryuuzeno/Downloads/Project/EvalScript/evalResult.py /Users/ouryuuzeno/Downloads/Project/ES/dev.out /Users/ouryuu
zeno/Downloads/Project/ES/dev.p3.out

#Entity in gold data: 255
#Entity in prediction: 505

#Correct Entity : 147
Entity  precision: 0.2911
Entity  recall: 0.5765
Entity  F: 0.3868

#Correct Sentiment : 89
Sentiment  precision: 0.1762
Sentiment  recall: 0.3490
Sentiment  F: 0.2342
(base) Swees-MacBook-Pro-2:~ ouryuuzeno$ python3 /Users/ouryuuzeno/Downloads/Project/EvalScript/evalResult.py /Users/ouryuuzeno/Downloads/Project/RU/dev.out /Users/ouryuu
zeno/Downloads/Project/RU/dev.p3.out

#Entity in gold data: 461
#Entity in prediction: 799

#Correct Entity : 232
Entity  precision: 0.2904
Entity  recall: 0.5033
Entity  F: 0.3683

#Correct Sentiment : 127
Sentiment  precision: 0.1589
Sentiment  recall: 0.2755
Sentiment  F: 0.2016
```

Part 4

Design Approach

We used the **Naive Bayes** algorithm to classify the words into the 7 labels.

Step 1:
For our training dataset, we will count the number of times a word and label appear together, the number of times each label appears and the total number of lines.

Step 2:
Using the values found in Step 1, we will find the probability that a word and label appear together (in Equation 1), the probability of a label appearing (in Equation 2) and the label that has the highest probability of appearing (in Equation 3).

$$P(word\ and\ label\ ) \ = \ \frac{No.\ of\ times\ a\ word\ appears\ with\ the\ label}{Total\ no.\ of\ lines} \ - \ Equation\ 1$$

$$P(label) \ = \ \frac{No.\ of\ times\ label\ appears}{Total\ no.\ of\ lines} \ - \ Equation\ 2$$

$$Label\ * \ = \ max_{label}\ [\ No.\ of\ times\ label\ appears\ ] \ - \ Equation\ 3$$

Since each word can have multiple labels, the predicted label we use for our model will be the label with the highest probability of appearing.

Step 3:
Find the probability of a label appearing given a word using the probabilities found in Step 2.

$$P(label\ |\ word) \ = \ \frac{P(word\ and\ label)}{P(label)}$$

Step 4:
Generate labels for test dataset based on the trained probabilities in Step 3. If a word did not appear in the training dataset, the word will be assigned to the label with the highest probability of appearing in the training dataset as found in Step 2. Then, write all the output to the output file.

Results

```
chuaqibao@Chuas-MacBook-Pro Project_Submission 2 % pyt
hon3 EvalScript/evalResult.py ES/dev.out ES/dev.p4.out


#Entity in gold data: 255
#Entity in prediction: 279

#Correct Entity : 121
Entity  precision: 0.4337
Entity  recall: 0.4745
Entity  F: 0.4532

#Correct Sentiment : 63
Sentiment  precision: 0.2258
Sentiment  recall: 0.2471
Sentiment  F: 0.2360
```

```
chuaqibao@Chuas-MacBook-Pro Project_Submission 2 % pyt
hon3 EvalScript/evalResult.py RU/dev.out RU/dev.p4.out


#Entity in gold data: 461
#Entity in prediction: 438

#Correct Entity : 222
Entity  precision: 0.5068
Entity  recall: 0.4816
Entity  F: 0.4939

#Correct Sentiment : 131
Sentiment  precision: 0.2991
Sentiment  recall: 0.2842
Sentiment  F: 0.2914
```