

# 1D Group Project Report 50.007 Machine Learning SUTD 2021 ISTD

#### Group 4

Ma Yuchen	1004519
Chung Wah Kit	1004103
James Raphael Tiovalen	1004555

# Table of Contents

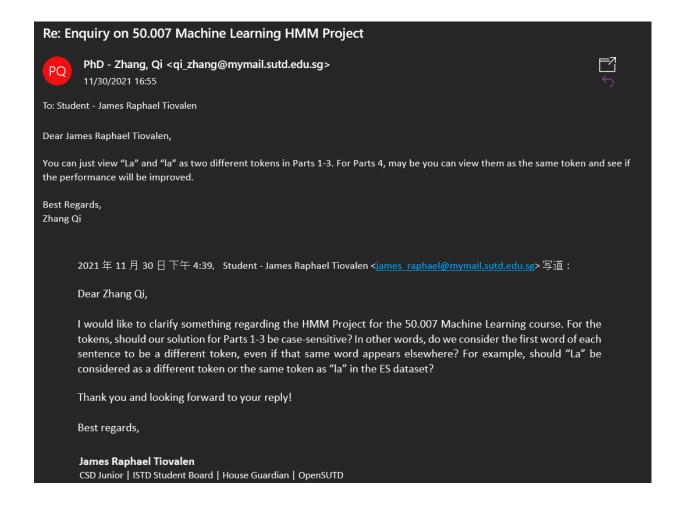
Table of Contents	2
Parts 1-3	3
Part 4	8
General Comments	6

#### Parts 1-3

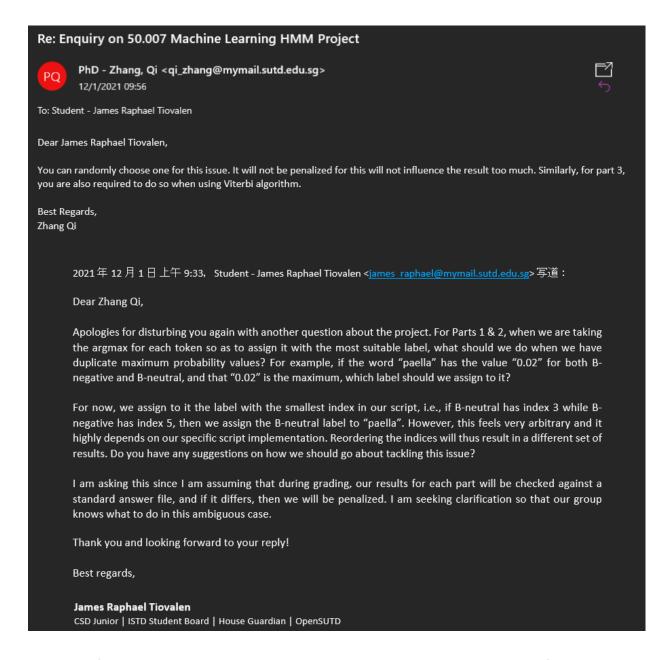
For parts 1-3, we simply followed the project instructions document closely as our general approach. We utilized the NumPy library to aid with all of our calculations. Using NumPy, we form matrices similar to the ones specified in the HMM classes. This is applicable and relevant for parts 1-3.

When reading the input files, we simply split each line by the rightmost whitespace character to obtain the word-tag pairs.

Our approach for different letter cases would be to treat them differently, as instructed in this email correspondence with the Teaching Assistant, Zhang Qi:



One special case to take care of would be when there are multiple tags with the same maximum value of e(x|y), and thus are all possible valid candidates as the argmax tag. After our correspondence, attached here:



We decided to randomly select between those valid tags, using the sum of our student IDs as the input seed to NumPy to ensure reproducible behavior.

In the case whereby we encounter numerical underflow issues in parts 2 and 3, we utilize the log-likelihood

and maximize that instead of the normal likelihood instead (and disable any warnings regarding np.log(0) since they will possess the minimum value possible of -np.inf anyway).

For part 3, the steps of our algorithm are:

- Execute the Viterbi algorithm
- Sort the order of possible tags in descending order with respect to the value of p(x\_1, ..., x\_n, y\_1, ..., y\_n)
- Select the top 5 tags
- Select the 5-th best (i.e., last one) out of the top 5

For Part 1, the output precision, recall, and F scores for both the ES and RU datasets are:

<u>ES</u>	<u>RU</u>
<pre># Entity in gold data: 255</pre>	<pre># Entity in gold data: 461</pre>
# Entity in prediction: 1734	# Entity in prediction: 2089
# of Correct Entity: 205 Entity precision: 0.1182 Entity recall: 0.8039 Entity F: 0.2061	<pre># of Correct Entity: 335 Entity precision: 0.1604 Entity recall: 0.7267 Entity F: 0.2627</pre>
<pre># of Correct Sentiment: 113</pre>	<pre># of Correct Sentiment: 136</pre>
Sentiment precision: 0.0652	Sentiment precision: 0.0651
Sentiment recall: 0.4431 Sentiment F: 0.1136	Sentiment recall: 0.2950 Sentiment F: 0.1067

For Part 2, the output precision, recall, and F scores for both the ES and RU datasets are:

<pre>ES # Entity in gold data: 255 # Entity in prediction: 686</pre>	<u><b>RU</b></u> # Entity in gold data: 461 # Entity in prediction: 575
# of Correct Entity: 131 Entity precision: 0.1910 Entity recall: 0.5137 Entity F: 0.2784	# of Correct Entity: 223 Entity precision: 0.3878 Entity recall: 0.4837 Entity F: 0.4305
<pre># of Correct Sentiment: 104 Sentiment precision: 0.1516 Sentiment recall: 0.4078 Sentiment F: 0.2210</pre>	<pre># of Correct Sentiment: 145 Sentiment precision: 0.2522 Sentiment recall: 0.3145 Sentiment F: 0.2799</pre>

For Part 3, the output precision, recall, and F scores for both the ES and RU datasets are:

ES RU # Entity in gold data: # Entity in gold data: 255 461 # Entity in prediction: # Entity in prediction: 556 535 # of Correct Entity: 219 # of Correct Entity: 131 Entity precision: 0.2356 Entity precision: 0.4093 Entity recall: 0.5137 Entity recall: 0.4751 Entity F: 0.3231 Entity F: 0.4398 # of Correct Sentiment: # of Correct Sentiment: 104 144 Sentiment precision: Sentiment precision: 0.1871 0.2692 Sentiment recall: 0.4078 Sentiment recall: 0.3124 Sentiment F: 0.2565 Sentiment F: 0.2892

### Part 4

For part 4, after reading into the literature, we decided to implement a unidirectional LSTM model using PyTorch. We also decided to convert all training words into lowercase as it ever-so-slightly improves our results.

The overall structure of our LSTM model is as such:

```
LSTMTagger((word_embeddings): Embedding(4805, 16)
  (lstm): LSTM(16, 16, num_layers=4)
  (hidden2tag): Linear(in_features=16, out_features=7, bias=True))
```

For Part 4, the <u>best</u> output precision, recall, and F scores for both the ES and RU datasets so far are:

<u>ES</u>	RU
Number of Epochs: 25	Number of Epochs: 45
# Entity in gold data:	# Entity in gold data:
255	461
# Entity in prediction:	# Entity in prediction:
182	329
# of Correct Entity: 125	# of Correct Entity: 208
Entity precision: 0.6868	Entity precision: 0.6322
Entity recall: 0.4902	Entity recall: 0.4512
Entity F: 0.5721	Entity F: 0.5266
# of Correct Sentiment:	<pre># of Correct Sentiment:</pre>
101	153
Sentiment precision:	Sentiment precision:
0.5549	0.4650
Sentiment recall: 0.3961	Sentiment recall: 0.3319
Sentiment F: 0.4622	Sentiment F: 0.3873

Future improvements could potentially be to improve the unidirectional LSTM by changing it into a bidirectional one, even though it would require more time and resources to train due to the increase in number of parameters. This trade-off in amount of training time and resources in exchange for better F-scores of the model is prevalent in the machine learning world, as models get larger and larger. Other potentially promising models include a BERT-BiLSTM-CRF model, <a href="OpenAI's GPT-3">OpenAI's GPT-3</a>, or even the latest Google's GLaM model.

## General Comments

To execute or modify any parts of our code, simply open up the Jupyter Notebook files `parts\_1\_to\_3.ipynb` and `part\_4.ipynb`.

Our GitHub repository is available here: <a href="https://github.com/jamestiotio/humu\_humu">https://github.com/jamestiotio/humu\_humu</a>