



SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

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	10

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:

Re: Enquiry on 50.007 Machine Learning HMM Project



PhD - Zhang, Qi <qi_zhang@mymail.sutd.edu.sg>
11/30/2021 16:55



To: Student - James Raphael Tiovalen

Dear James Raphael Tiovalen,

You can just view "La" and "la" as two different tokens in Parts 1-3. For Parts 4, may be you can view them as the same token and see if the performance will be improved.

Best Regards,
Zhang Qi

2021 年 11 月 30 日 下午 4:39, Student - James Raphael Tiovalen <james_raphael@mymail.sutd.edu.sg> 写道:

Dear Zhang Qi,

I would like to clarify something regarding the HMM Project for the 50.007 Machine Learning course. For the tokens, should our solution for Parts 1-3 be case-sensitive? In other words, do we consider the first word of each sentence to be a different token, even if that same word appears elsewhere? For example, should "La" be considered as a different token or the same token as "la" in the ES dataset?


Thank you and looking forward to your reply!

Best regards,


James Raphael Tiovalen
CSD Junior | ISTD Student Board | House Guardian | OpenSUTD

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:

Re: Enquiry on 50.007 Machine Learning HMM Project



PhD - Zhang, Qi <qi_zhang@mymail.sutd.edu.sg>
12/1/2021 09:56



To: Student - James Raphael Tiovalen

Dear James Raphael Tiovalen,

You can randomly choose one for this issue. It will not be penalized for this will not influence the result too much. Similarly, for part 3, you are also required to do so when using Viterbi algorithm.

Best Regards,
Zhang Qi

2021 年 12 月 1 日 上午 9:33, Student - James Raphael Tiovalen <james_raphael@mymail.sutd.edu.sg> 写道:

Dear Zhang Qi,

Apologies for disturbing you again with another question about the project. For Parts 1 & 2, when we are taking the argmax for each token so as to assign it with the most suitable label, what should we do when we have duplicate maximum probability values? For example, if the word "paella" has the value "0.02" for both B-negative and B-neutral, and that "0.02" is the maximum, which label should we assign to it?

For now, we assign to it the label with the smallest index in our script, i.e., if B-neutral has index 3 while B-negative has index 5, then we assign the B-neutral label to "paella". However, this feels very arbitrary and it highly depends on our specific script implementation. Reordering the indices will thus result in a different set of results. Do you have any suggestions on how we should go about tackling this issue?

I am asking this since I am assuming that during grading, our results for each part will be checked against a standard answer file, and if it differs, then we will be penalized. I am seeking clarification so that our group knows what to do in this ambiguous case.

Thank you and looking forward to your reply!

Best regards,

James Raphael Tiovalen
CSD Junior | ISTD Student Board | House Guardian | OpenSUTD

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 $p(x_1, \dots, x_n, y_1, \dots, y_n)$
- 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:

ES

Entity in gold data:
255
Entity in prediction:
1734

of Correct Entity: 205
Entity precision: 0.1182
Entity recall: 0.8039
Entity F: 0.2061

of Correct Sentiment:
113
Sentiment precision:
0.0652
Sentiment recall: 0.4431
Sentiment F: 0.1136

RU

Entity in gold data:
461
Entity in prediction:
2089

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

of Correct Sentiment:
136
Sentiment precision:
0.0651
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:

ES

Entity in gold data:
255
Entity in prediction:
686

of Correct Entity: 131
Entity precision: 0.1910
Entity recall: 0.5137
Entity F: 0.2784

of Correct Sentiment:
104
Sentiment precision:
0.1516
Sentiment recall: 0.4078
Sentiment F: 0.2210

RU

Entity in gold data:
461
Entity in prediction:
575

of Correct Entity: 223
Entity precision: 0.3878
Entity recall: 0.4837
Entity F: 0.4305

of Correct Sentiment:
145
Sentiment precision:
0.2522
Sentiment recall: 0.3145
Sentiment F: 0.2799

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

ES

Entity in gold data:
255
Entity in prediction:
556

of Correct Entity: 131
Entity precision: 0.2356
Entity recall: 0.5137
Entity F: 0.3231

of Correct Sentiment:
104
Sentiment precision:
0.1871
Sentiment recall: 0.4078
Sentiment F: 0.2565

RU

Entity in gold data:
461
Entity in prediction:
535

of Correct Entity: 219
Entity precision: 0.4093
Entity recall: 0.4751
Entity F: 0.4398

of Correct Sentiment:
144
Sentiment precision:
0.2692
Sentiment recall: 0.3124
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 **best** output precision, recall, and F scores for both the ES and RU datasets so far are:

ES

Number of Epochs: 25
 # Entity in gold data:
 255
 # Entity in prediction:
 182

 # of Correct Entity: 125
 Entity precision: 0.6868
 Entity recall: 0.4902
 Entity F: 0.5721

 # of Correct Sentiment:
 101
 Sentiment precision:
 0.5549
 Sentiment recall: 0.3961
 Sentiment F: 0.4622

RU

Number of Epochs: 45
 # Entity in gold data:
 461
 # Entity in prediction:
 329

 # of Correct Entity: 208
 Entity precision: 0.6322
 Entity recall: 0.4512
 Entity F: 0.5266

 # of Correct Sentiment:
 153
 Sentiment precision:
 0.4650
 Sentiment recall: 0.3319
 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, [OpenAI's GPT-3](#), or even the [latest Google's GLaM model](#).

General Comments

To inspect, execute, or modify any parts of our code, simply open up the Jupyter Notebook files `parts_1_to_3.ipynb` and `part_4.ipynb`. Each cell in the 2 notebooks is executable as per the Jupyter Notebook standard.

Our GitHub repository is available here:
https://github.com/jamestiotio/humu_humu