



SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

50.007 MACHINE LEARNING PROJECT REPORT

Abel Lee Yang Yeow 1006085

Loh Jianyang John 1006360




















Chia Chun Mun 1005934

REPORT SUMMARIZATION

Goal: Build a sequence labelling system from such training data and then use the system to predict tag sequences for new sentences.

PROJECT DIRECTORY













Main Directory

 Data	13/8/2023 3:01 am	File folder	
 Part 1	13/8/2023 12:52 pm	Jupyter Source File	16 KB
 Part 1	13/8/2023 1:12 pm	PY File	9 KB
 Part 2	13/8/2023 12:53 pm	Jupyter Source File	22 KB
 Part 2	13/8/2023 1:12 pm	PY File	14 KB
 Part 2_Toy_Example	13/8/2023 12:53 pm	Jupyter Source File	173 KB
 Part 2_Toy_Example	13/8/2023 1:12 pm	PY File	12 KB
 Part 3	13/8/2023 12:57 pm	Jupyter Source File	30 KB
 Part 3	13/8/2023 1:12 pm	PY File	20 KB
 Part 3_allseq_Toy_Example	13/8/2023 12:57 pm	Jupyter Source File	68 KB
 Part 3_allseq_Toy_Example	13/8/2023 1:12 pm	PY File	9 KB
 Part 3_kbest_Toy_Example	13/8/2023 12:57 pm	Jupyter Source File	61 KB
 Part 3_kbest_Toy_Example	13/8/2023 1:12 pm	PY File	9 KB
 Part 4_dev	13/8/2023 1:08 pm	Jupyter Source File	25 KB
 Part 4_dev	13/8/2023 1:13 pm	PY File	16 KB
 Part 4_test	13/8/2023 1:08 pm	Jupyter Source File	23 KB
 Part 4_test	13/8/2023 1:13 pm	PY File	15 KB
 project_utils	11/8/2023 2:23 pm	PY File	5 KB
 README	13/8/2023 1:03 pm	Text Document	1 KB













\Data

 ES	13/8/2023 3:01 am	File folder
 RU	13/8/2023 3:02 am	File folder

\Data\ES

 .DS_Store	28/7/2023 9:41 pm	DS_STORE File	7 KB
 dev.in	28/7/2023 9:41 pm	IN File	23 KB
 dev	28/7/2023 9:41 pm	Wireshark capture file	34 KB
 dev.p1	13/8/2023 12:49 pm	Wireshark capture file	50 KB
 dev.p2	13/8/2023 12:52 pm	Wireshark capture file	43 KB
 dev.p3.2nd	13/8/2023 12:54 pm	Wireshark capture file	41 KB
 dev.p3.8th	13/8/2023 12:54 pm	Wireshark capture file	43 KB
 dev.p4	13/8/2023 12:58 pm	Wireshark capture file	39 KB
 evalResult	28/7/2023 10:50 pm	PY File	7 KB
 test.in	13/8/2023 1:14 am	IN File	24 KB
 test.p4	13/8/2023 1:01 pm	Wireshark capture file	41 KB
 train	28/7/2023 9:41 pm	File	241 KB

\Data\RU

 .DS_Store	28/7/2023 9:41 pm	DS_STORE File	7 KB
 dev.in	28/7/2023 9:41 pm	IN File	64 KB
 dev	28/7/2023 9:41 pm	Wireshark capture file	82 KB
 dev.p1	13/8/2023 12:49 pm	Wireshark capture file	101 KB
 dev.p2	13/8/2023 12:53 pm	Wireshark capture file	91 KB
 dev.p3.2nd	13/8/2023 12:56 pm	Wireshark capture file	92 KB
 dev.p3.8th	13/8/2023 12:56 pm	Wireshark capture file	93 KB
 dev.p4	13/8/2023 12:59 pm	Wireshark capture file	88 KB
 evalResult	28/7/2023 10:50 pm	PY File	7 KB
 test.in	13/8/2023 1:14 am	IN File	58 KB
 test.p4	13/8/2023 1:02 pm	Wireshark capture file	81 KB
 train	28/7/2023 9:41 pm	File	534 KB

Note:

- Both .py files and .ipynb files have been provided and are labelled according to the question part
- Outputs of project can be found in \Data\EU and \Data\RU
- project_utils.py is used to import functions that have already been written to prevent code repetition in Part 2 and Part 3

Instructions to run evaluation script

1. Navigate to the ES or RU folder
2. Enter the command `python evalResult.py dev.out <file to evaluate>`

Example:

```
(base) C:\Users\johnl>activate myenv

(myenv) C:\Users\johnl>cd C:\Users\johnl\Documents\Term 5\Machine Learning\Machine Learning Project

(myenv) C:\Users\johnl\Documents\Term 5\Machine Learning\Machine Learning Project>cd Data

(myenv) C:\Users\johnl\Documents\Term 5\Machine Learning\Machine Learning Project\Data>cd ES

(myenv) C:\Users\johnl\Documents\Term 5\Machine Learning\Machine Learning Project\Data\ES>python evalResult.py dev.out dev.p1.out

#Entity in gold data: 229
#Entity in prediction: 1466

#Correct Entity : 178
Entity precision: 0.1214
Entity recall: 0.7773
Entity F: 0.2100

#Correct Sentiment : 97
Sentiment precision: 0.0662
Sentiment recall: 0.4236
Sentiment F: 0.1145

(myenv) C:\Users\johnl\Documents\Term 5\Machine Learning\Machine Learning Project\Data\ES>
```

Part 1

Name of File: Part 1.py, Part 1.ipynb

Instructions on how to run the code: Restart and run all cells

Location of ES dev.p1.out: Data\ES\dev.p1

Location of RU dev.p1.out: Data\RU\dev.p1

Results of Part 1, Precision, Recall, F scores:

ES dev.p1.out

```
#Entity in gold data: 229
#Entity in prediction: 1466

#Correct Entity : 178
Entity precision: 0.1214
Entity recall: 0.7773
Entity F: 0.2100

#Correct Sentiment : 97
Sentiment precision: 0.0662
Sentiment recall: 0.4236
Sentiment F: 0.1145
```

Entity precision: 0.1214

Entity recall: 0.7773

Entity F: 0.2100

Sentiment precision: 0.0662

Sentiment recall: 0.4236

Sentiment F: 0.1145

RU dev.p1.out

```
#Entity in gold data: 389
#Entity in prediction: 1816

#Correct Entity : 266
Entity precision: 0.1465
Entity recall: 0.6838
Entity F: 0.2413

#Correct Sentiment : 129
Sentiment precision: 0.0710
Sentiment recall: 0.3316
Sentiment F: 0.1170
```

Entity precision: 0.1465

Entity recall: 0.6838

Entity F: 0.2413

Sentiment precision: 0.00710

Sentiment recall: 0.3316

Sentiment F: 0.1170

Part 2

Name of File: Part 2.py, Part 2.ipynb

Additional File: Part 2_Toy_Example.py, Part 2_Toy_Example.ipynb (File that demonstrates the Viterbi code being run step by step on 1 sentence to provide more granularity)

Instructions on how to run the code: Restart and run all cells

Location of ES dev.p2.out: Data\ES\dev.p2

Location of RU dev.p2.out: Data\RU\dev.p2

Results of Part 2, Precision, Recall, F scores:

ES dev.p2.out

```
#Entity in gold data: 229
#Entity in prediction: 542

#Correct Entity : 134
Entity precision: 0.2472
Entity recall: 0.5852
Entity F: 0.3476

#Correct Sentiment : 97
Sentiment precision: 0.1790
Sentiment recall: 0.4236
Sentiment F: 0.2516
```

Entity precision: 0.2472

Entity recall: 0.5852

Entity F: 0.3476

Sentiment precision: 0.1790

Sentiment recall: 0.4236

Sentiment F: 0.2516

RU dev.p2.out

```
#Entity in gold data: 389
#Entity in prediction: 484

#Correct Entity : 188
Entity precision: 0.3884
Entity recall: 0.4833
Entity F: 0.4307

#Correct Sentiment : 129
Sentiment precision: 0.2665
Sentiment recall: 0.3316
Sentiment F: 0.2955
```

Entity precision: 0.3884

Entity recall: 0.4833

Entity F: 0.4307

Sentiment precision: 0.2665

Sentiment recall: 0.3316

Sentiment F: 0.2955

Part 3

Name of File: Part 3.py, Part 3.ipynb

Additional Files:

1. Part 3_kbest_Toy_Example.py, Part 3_kbest_Toy_Example.ipynb (File that demonstrates the modified Viterbi code to find k-best sequences using beam search in the forward pass, keeping only the k best sequences after every position is done, being run step by step on 1 sentence to provide more granularity)
2. Part 3_allseq_Toy_Example.py, Part 3_allseq_Toy_Example.ipynb (File that demonstrates the modified Viterbi code to find k-best sequences using beam search in the forward pass, keeping all sequences after every position is done, being run step by step on 1 sentence to provide more granularity)

Instructions on how to run the code: Restart and run all cells

Location of ES dev.p3.2nd.out: Data\ES\dev.p3.2nd

Location of ES dev.p3.8th.out: Data\ES\dev.p3.8th

Location of RU dev.p3.2nd.out: Data\RU\dev.p3.2nd

Location of RU dev.p3.8th.out: Data\RU\dev.p3.8th

Results of Part 3, Precision, Recall, F scores:

ES dev.p3.2nd.out:

```
#Entity in gold data: 229
#Entity in prediction: 454

#Correct Entity : 119
Entity precision: 0.2621
Entity recall: 0.5197
Entity F: 0.3485

#Correct Sentiment : 70
Sentiment precision: 0.1542
Sentiment recall: 0.3057
Sentiment F: 0.2050
```

Entity precision: 0.2621

Entity recall: 0.5197

Entity F: 0.3485

Sentiment precision: 0.1542

Sentiment recall: 0.3057

Sentiment F: 0.2050

ES dev.p3.8th.out:

```
#Entity in gold data: 229
#Entity in prediction: 539

#Correct Entity : 106
Entity precision: 0.1967
Entity recall: 0.4629
Entity F: 0.2760

#Correct Sentiment : 63
Sentiment precision: 0.1169
Sentiment recall: 0.2751
Sentiment F: 0.1641
```

Entity precision: 0.1967

Entity recall: 0.4629

Entity F: 0.2760

Sentiment precision: 0.1169

Sentiment recall: 0.2751

Sentiment F: 0.1641

RU dev.p3.2nd.out:

```
#Entity in gold data: 389
#Entity in prediction: 677

#Correct Entity : 198
Entity precision: 0.2925
Entity recall: 0.5090
Entity F: 0.3715

#Correct Sentiment : 123
Sentiment precision: 0.1817
Sentiment recall: 0.3162
Sentiment F: 0.2308
```

Entity precision: 0.2925

Entity recall: 0.5090

Entity F: 0.3715

Sentiment precision: 0.1817

Sentiment recall: 0.3162

Sentiment F: 0.2308

RU dev.p3.8th.out:

```
#Entity in gold data: 389
#Entity in prediction: 779

#Correct Entity : 176
Entity precision: 0.2259
Entity recall: 0.4524
Entity F: 0.3014

#Correct Sentiment : 101
Sentiment precision: 0.1297
Sentiment recall: 0.2596
Sentiment F: 0.1729
```

Entity precision: 0.2259

Entity recall: 0.4524

Entity F: 0.3014

Sentiment precision: 0.1297

Sentiment recall: 0.2596

Sentiment F: 0.1729

Explanation of the steps of the algorithm

Modified Viterbi Algorithm to compute k-best sequences using beam search in the forward pass, keeping only the k-best sequences after each position has been iterated through

In the initialisation step of the algorithm, we initialise a max heap containing the score of the start state and the start state itself. [(1, ['START'])]. This can be seen more generally as:

[(score of potential sequence, [sequence path of states])]

We then iterate through every position in the sentence, from the position of the first word to the position of the last word. For each position, we iterate through all k-best sequences (max k sequences) we currently have in order to extend the sequence:

[(score of potential extended sequence, [extended sequence path of states])].

We loop through all states in the current position to perform the extension for all best sequences.

For each best sequence:

For each possible state in a position:

1. Calculate the extended sequence score: score of the best sequence * transition probability of the latest state in the best sequence to this state * emission probability of the observation from this state
2. Extend the sequence path with this state
3. Save [(score of potential extended sequence, [extended sequence path of states])] into a heap

After performing this extension for all best sequences, we perform pruning on the max heap where we keep only the top-k sequences in order to achieve efficiency. To reiterate, the scores are computed using joint probability, the product of emissions and transitions of that path.

In the final step, we loop through all the best sequences to calculate its sequence score by considering its transition probability to the 'STOP' state.

For each best sequence:

1. Calculate the extended sequence score: score of the best sequence * transition probability of the latest state in the best sequence to 'STOP' state
2. Extend the sequence path with 'STOP' state
3. Save [(score of potential extended sequence, [extended sequence path of states]) into a heap

After performing this final extension for all best sequences, we perform pruning on the max heap where we keep only the top-k sequences.

In the event that there are no sequences, *which is possible if, at some point, the latest observation of the best sequence exists in the training set but its emission probability does not exist, leading to these sequences getting dropped*, we call the modified Viterbi Algorithm that also computes k-best sequences using beam search in the forward pass, but keeps all sequences after each position has been iterated through. This expands the beam and is a greedy algorithm as it considers all possible sequences, unlike this algorithm which only considers the k-best sequences after every iteration.

The downside to the algorithm that keeps only the k-best sequences after each iteration is that it may drop optimal sequences in earlier time steps/positions as optimal sequences may not achieve a high sequence score in earlier time steps/positions but may make up for it in later time steps/positions. The upside to this algorithm is that it is faster.

Finally, if less than k sequences are found, we pad the sequences with the worst performing sequence until we have k sequences.

Part 4

Name of File:

Part 4_dev.py, Part 4_dev.ipynb (ran on dev.in)

Part 4_test.py, Part 4_test.ipynb (ran on test.in)

Instructions on how to run the code: Restart and run all cells

Location of ES dev.p4.out: Data\ES\dev.p4

Location of RU dev.p4.out: Data\RU\dev.p4

Location of ES test.p4.out: Data\ES\test.p4

Location of RU test.p4.out: Data\RU\test.p4

Results of Part 4, Precision, Recall, F scores:

EU dev.p4.out:

```
#Entity in gold data: 229
#Entity in prediction: 217

#Correct Entity : 138
Entity precision: 0.6359
Entity recall: 0.6026
Entity F: 0.6188

#Correct Sentiment : 108
Sentiment precision: 0.4977
Sentiment recall: 0.4716
Sentiment F: 0.4843
```

Entity precision: 0.6359

Entity recall: 0.6026

Entity F: 0.6188

Sentiment precision: 0.4977

Sentiment recall: 0.4716

Sentiment F: 0.4843

RU dev.p4.out:

```
#Entity in gold data: 389
#Entity in prediction: 312

#Correct Entity : 189
Entity precision: 0.6058
Entity recall: 0.4859
Entity F: 0.5392

#Correct Sentiment : 136
Sentiment precision: 0.4359
Sentiment recall: 0.3496
Sentiment F: 0.3880
```

Entity precision: 0.6058

Entity recall: 0.4859

Entity F: 0.5392

Sentiment precision: 0.4359

Sentiment recall: 0.3496

Sentiment F: 0.3880

Approach that we used to improve results: Laplace Smoothing

Explanation:

Using Laplace Smoothing, we better handled the problem of emission probability for unknown words (*unknown observation, state*) using the following formula:

$$\text{Laplace Smoothing} = \frac{kvalue + \alpha}{(number\ of\ times\ state\ occurs + kvalue) + \alpha * vocabulary\ size}$$

To avoid estimating any emission probabilities to be zero for unknown words, we can adjust alpha to increase the emission probability even if the word is unknown.

Experimentation of applying Laplace Smoothing to trained and unknown words using $\alpha=1$ (recommended)

1. Laplace smoothing to emission for unknown words only (Left is ES & Right is RU)

#Entity in gold data: 229 #Entity in prediction: 217	#Entity in gold data: 389 #Entity in prediction: 312
#Correct Entity : 138 Entity precision: 0.6359 Entity recall: 0.6026 Entity F: 0.6188	#Correct Entity : 189 Entity precision: 0.6058 Entity recall: 0.4859 Entity F: 0.5392
#Correct Sentiment : 108 Sentiment precision: 0.4977 Sentiment recall: 0.4716 Sentiment F: 0.4843	#Correct Sentiment : 136 Sentiment precision: 0.4359 Sentiment recall: 0.3496 Sentiment F: 0.3880

2. Laplace smoothing to emission for normal words only (Left is ES & Right is RU)

#Entity in gold data: 229 #Entity in prediction: 489	#Entity in gold data: 389 #Entity in prediction: 304
#Correct Entity : 93 Entity precision: 0.1902 Entity recall: 0.4061 Entity F: 0.2591	#Correct Entity : 111 Entity precision: 0.3651 Entity recall: 0.2853 Entity F: 0.3203
#Correct Sentiment : 68 Sentiment precision: 0.1391 Sentiment recall: 0.2969 Sentiment F: 0.1894	#Correct Sentiment : 75 Sentiment precision: 0.2467 Sentiment recall: 0.1928 Sentiment F: 0.2165

3. Laplace smoothing to emission for both unknown and normal words (Left is ES & Right is RU)

#Entity in gold data: 229 #Entity in prediction: 118	#Entity in gold data: 389 #Entity in prediction: 152
#Correct Entity : 85 Entity precision: 0.7203 Entity recall: 0.3712 Entity F: 0.4899	#Correct Entity : 99 Entity precision: 0.6513 Entity recall: 0.2545 Entity F: 0.3660
#Correct Sentiment : 70 Sentiment precision: 0.5932 Sentiment recall: 0.3057 Sentiment F: 0.4035	#Correct Sentiment : 76 Sentiment precision: 0.5000 Sentiment recall: 0.1954 Sentiment F: 0.2810

After experimentation and analysis, we arrived at the following conclusion:

Applied Laplace Smoothing with alpha=1 to	ES data	RU data
Unknown words	Improved	Improved
Normal words	De-proved	De-proved
Both unknown and normal words	Improved	De-proved

We concluded that applying laplace smoothing to unknown words was the best approach.

We then tried adjusting the alpha values from 0.9 to 0.1 in step intervals of 0.1 and repeated our experiments

Applied Laplace Smoothing with different alpha values from 0.9 to 0.1	ES data	RU data
Unknown words	Trend: De-proved	Trend: De-proved
Both unknown and normal words	Trend: Improved	Trend: Improved

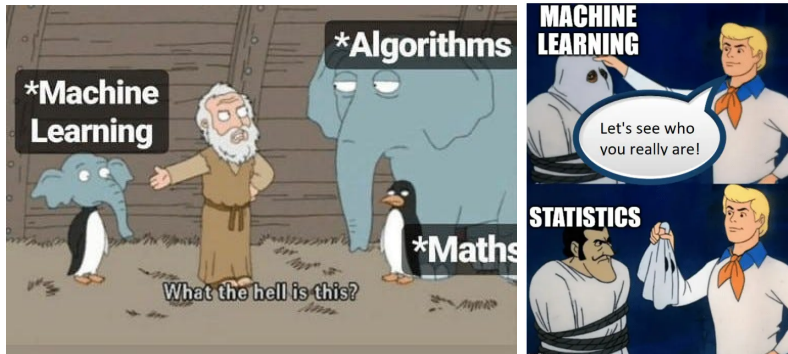
In the case of Laplace smoothing for both unknown and normal words, the F-scores keeps increasing till alpha=0.1, however, it still loses out to the result when we apply Laplace smoothing to unknown words only.

#Entity in gold data: 229 #Entity in prediction: 227 #Correct Entity : 124 Entity precision: 0.5463 Entity recall: 0.5415 Entity F: 0.5439 #Correct Sentiment : 99 Sentiment precision: 0.4361 Sentiment recall: 0.4323 Sentiment F: 0.4342	#Entity in gold data: 389 #Entity in prediction: 397 #Correct Entity : 175 Entity precision: 0.4408 Entity recall: 0.4499 Entity F: 0.4453 #Correct Sentiment : 131 Sentiment precision: 0.3300 Sentiment recall: 0.3368 Sentiment F: 0.3333	VS	#Entity in gold data: 229 #Entity in prediction: 217 #Correct Entity : 138 Entity precision: 0.6359 Entity recall: 0.6026 Entity F: 0.6188 #Correct Sentiment : 108 Sentiment precision: 0.4977 Sentiment recall: 0.4716 Sentiment F: 0.4843	#Entity in gold data: 389 #Entity in prediction: 312 #Correct Entity : 189 Entity precision: 0.6058 Entity recall: 0.4859 Entity F: 0.5392 #Correct Sentiment : 136 Sentiment precision: 0.4359 Sentiment recall: 0.3496 Sentiment F: 0.3880
--	---	----	---	---

Left image: Laplace smoothing on both unknown and normal words with alpha=0.1

Right image: Laplace smoothing on both unknown and normal words only with alpha=1

Conclusion: We concluded to use the Laplace smoothing with $\alpha=1$ to the emission of the unknown words which will give us the best score and used this to calculate the new emission probabilities of unknown words.



References

Beam Search.

<https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24>

Laplace Smoothing.

<https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece>