SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

01.112 Machine Learning (2017)

Project: Sentiment Analysis

Joshua Lim Wen Yao (1001509)
Burindy Yeo (1001655)
Pinardy Yang (1001520)

Part 2

File for part 2: sentiment_analysis_part2.ipynb

For this project, we are using pandas and numpy to manipulate the data. By using Jupyter notebook as our editor, we can code the algorithms and display results on the notebook itself. This is beneficial for us for testing.

Example: 'SG' training set:

State	Observation
B-negative	1299
B-neutral	5722
B-positive	2613
I-negative	443
I-neutral	5272
I-positive	1653
O	91753
Start	7094

Instructions

1) Open the .ipynb file on Jupyter notebook
2) The code can be run by clicking on 'Kernel' and clicking 'Restart & Run All'
3) To change the language of choice, go to the last cell of the notebook and change the string contained by the variable to your language of choice
4) Run from terminal: python evalResult.py dev.out dev.p2.out

```
In [12]: pd.options.mode.chained_assignment = None  # Turn off warning message

         '''
         Languages: 'CN' , 'EN', 'FR', 'SG'
         Change the language below accordingly
         '''
         language = 'EN'
         __main__(language)
```

Part 2 Results

SG

#Entity in gold data: 1382
#Entity in prediction: 6599

#Correct Entity : 794
Entity precision: 0.1203
Entity recall: 0.5745
Entity F: 0.1990

#Correct Sentiment : 315
Sentiment precision: 0.0477
Sentiment recall: 0.2279
Sentiment F: 0.0789

EN

#Entity in gold data: 226
#Entity in prediction: 1207

#Correct Entity : 165
Entity precision: 0.1367
Entity recall: 0.7301
Entity F: 0.2303

#Correct Sentiment : 71
Sentiment precision: 0.0588
Sentiment recall: 0.3142
Sentiment F: 0.0991

FR

#Entity in gold data: 223
#Entity in prediction: 1149

#Correct Entity : 182
Entity precision: 0.1584
Entity recall: 0.8161
Entity F: 0.2653

#Correct Sentiment : 68
Sentiment precision: 0.0592
Sentiment recall: 0.3049
Sentiment F: 0.0991

CN

#Entity in gold data: 362
#Entity in prediction: 3318

#Correct Entity : 183
Entity precision: 0.0552
Entity recall: 0.5055
Entity F: 0.0995

#Correct Sentiment : 57
Sentiment precision: 0.0172
Sentiment recall: 0.1575
Sentiment F: 0.0310

Part 3 – Viterbi Algorithm

File for part 3: sentiment_analysis_part3.ipynb

We continue to use pandas and numpy to handle and manipulate the data. However, we did conversions to lists and dictionaries for the computation itself to speed up the time taken to train.

We used log to handle the numerical underflow issue.

Instructions:

1) Open the .ipynb file on Jupyter notebook
2) The code can be run by clicking on 'Kernel' and clicking 'Restart & Run All'
3) To change the language of choice, go to the last cell of the notebook and change the string contained by the variable to your language of choice
4) Run from terminal: python evalResult.py dev.out dev.p3.out

In [10]:
```
def __main__(lang):
    np.seterr(divide='ignore') # Ignore log zero warnings
    df = createDf(lang,'/train')
    df_test = createDfDevin(lang,'/dev.in')

    predictions = Predict(df, df_test)
    with open('%s/dev.p3.out'%lang, 'w',encoding='utf8') as f:
        f.write(''.join(predictions))
        print("\nPrediction complete. File is saved")

'''
Languages: 'CN' , 'EN', 'FR', 'SG'
Change the language below accordingly
'''
language = 'EN'
__main__(language)
```

Part 3 Results

EN

#Entity in gold data: 226
#Entity in prediction: 175

#Correct Entity : 104
Entity precision: 0.5943
Entity recall: 0.4602
Entity F: 0.5187

#Correct Sentiment : 64
Sentiment precision: 0.3657
Sentiment recall: 0.2832
Sentiment F: 0.3192

FR

#Entity in gold data: 223
#Entity in prediction: 166

#Correct Entity : 112
Entity precision: 0.6747
Entity recall: 0.5022
Entity F: 0.5758

#Correct Sentiment : 72
Sentiment precision: 0.4337
Sentiment recall: 0.3229
Sentiment F: 0.3702

CN

#Entity in gold data: 362
#Entity in prediction: 228

#Correct Entity : 65
Entity precision: 0.2851
Entity recall: 0.1796
Entity F: 0.2203

#Correct Sentiment : 47
Sentiment precision: 0.2061
Sentiment recall: 0.1298
Sentiment F: 0.1593

SG

#Entity in gold data: 1382
#Entity in prediction: 731

#Correct Entity : 386
Entity precision: 0.5280
Entity recall: 0.2793
Entity F: 0.3654

#Correct Sentiment : 244
Sentiment precision: 0.3338
Sentiment recall: 0.1766
Sentiment F: 0.2310

Part 4 – Alternative max-marginal decoding

File for part 4: sentiment_analysis_part4.ipynb

We continue to use pandas and numpy to handle and manipulate the data.

Instructions:

1) Open the .ipynb file on Jupyter notebook
2) The code can be run by clicking on 'Kernel' and clicking 'Restart & Run All'
3) To change the language of choice, go to the last cell of the notebook and change the string contained by the variable to your language of choice
4) Run from terminal: python evalResult.py dev.out dev.p4.out

```
In [12]: def __main__(lang):
             np.seterr(divide='ignore')#Ignore log zero warnings
             df = createDf(lang,'/train')
             df_test = createDfDevin(lang,'/dev.in')

             predictions = Predict(df, df_test)
             with open('%s/dev.p4.out'%lang, 'w',encoding='utf8') as f:
                 f.write(''.join(predictions))
                 print("Saved.")

         '''
         Languages: 'CN' , 'EN', 'FR', 'SG'
         Change the language below accordingly
         '''
         language = 'CN'
         __main__(language)
```

Part 4 Results

SG

#Entity in gold data: 1382
#Entity in prediction: 775

#Correct Entity : 391
Entity precision: 0.5045
Entity recall: 0.2829
Entity F: 0.3625

#Correct Sentiment : 256
Sentiment precision: 0.3303
Sentiment recall: 0.1852
Sentiment F: 0.2374

EN

#Entity in gold data: 226
#Entity in prediction: 175

#Correct Entity : 108
Entity precision: 0.6171
Entity recall: 0.4779
Entity F: 0.5387

#Correct Sentiment : 69
Sentiment precision: 0.3943
Sentiment recall: 0.3053
Sentiment F: 0.3441

CN

#Entity in gold data: 362
#Entity in prediction: 261

#Correct Entity : 68
Entity precision: 0.2605
Entity recall: 0.1878
Entity F: 0.2183

#Correct Sentiment : 47
Sentiment precision: 0.1801
Sentiment recall: 0.1298
Sentiment F: 0.1509

FR

#Entity in gold data: 223
#Entity in prediction: 173

#Correct Entity : 113
Entity precision: 0.6532
Entity recall: 0.5067
Entity F: 0.5707

#Correct Sentiment : 73
Sentiment precision: 0.4220
Sentiment recall: 0.3274
Sentiment F: 0.3687

Part 5 – Challenge

File for part 5: sentiment_analysis_part5.ipynb

In this challenge, we will be implementing the Perceptron algorithm modified for Hidden Markov Model. The algorithm is referenced from Michael Collins's paper: http://www.aclweb.org/anthology/W02-1001

In this paper, parameter-estimation algorithms based on the perceptron algorithm are described. The paper concentrates on tagging problems, just like in this project.

Instructions:

1) Open the .ipynb file on Jupyter notebook
2) The code can be run by clicking on 'Kernel' and clicking 'Restart & Run All'
3) To change the language of choice, go to the last cell of the notebook and change the string contained by the variable to your language of choice
4) Run from terminal: python evalResult.py dev.out dev.p5.out

```
In [*]: def __main__(language):
            global_weights, global_trigram, predictions, pred_list = TrainPerceptron(language)

        language = 'EN'
        __main__(language)
```

Parameter Estimation

DISCLAIMER*: the following explanations are key points taken from Michael Collins's paper found in the reference.*

In a trigram HMM tagger, each trigram of tags and each tag/word pair have associated parameters. We write the parameter We write the parameter associated with a trigram <x, y, z> as $\alpha_{x,y,z}$, and the parameter associated with a tag/word pair (t,w) as $\alpha_{t,w}$. A common approach is to take the parameters to be estimates of conditional probabilities:

$\alpha_{x,y,z} = \log P(z \mid x; y)$,
$\alpha_{t,w} = \log P(w \mid t)$.

Algorithm:

1) Choose a parameter T defining the number of iterations over the training set.
2) Initially set all paramters $\alpha_{x,y,z}$ and $\alpha_{t,w}$ to be zero
3) For t = 1 … T, i = 1 … n, use the Viterbi Algorithm to find the best tagged sequence for sentence under the current parameter settings

For example, if the ith tag sentence in training data is:

D N V D N
The man saw the dog

and under the current parameter settings, the highest scoring tag sequence is:

D N N D N
The man saw the dog

Then the parameter update will add 1 to the parameters $\alpha_{D,N,V}$, $\alpha_{N,V,D}$, $\alpha_{V,D,N}$, $\alpha_{V,saw}$ and subtract 1 from the parameters $\alpha_{D,N,N}$, $\alpha_{N,N,D}$, $\alpha_{N,D,N}$, $\alpha_{N,saw}$ This has the effect of increasing the parameter values for features which were "missing" from the proposed sequence $z_{[1:ni]}$, and down weighting parameter values for "incorrect" features in the sequence. If the proposed tag sequence is correct, no changes are made to the parameter values

Obtain best scores:

To obtain the best scores, we try different learning rates and different number of iterations to see which combination gives us the best F score. The tables below illustrate which combinations provide the best F scores from using the modified Perceptron algorithm (without any conversion of the training set and dev.in to lowercase yet)

Given the three parameters iterations, word_lr and trigram_lr, we fixed 2 of the parameters while adjusting 1 to determine the best value for the variable. We repeated this for all combinations. After we converted the data to lowercase and set k = 2, we performed the above experimentation again and got the following results below.

EN Lower Case UNK2 – Viterbi only (No perceptron)				
Iterations	**Word_lr**	**Trigram_lr**	**Entity F**	**Sentiment F**
1	0	0	0.54	0.345

Fix Iteration, Trigram Learning Rate				
Iterations	**Word_lr**	**Trigram_lr**	**Entity F**	**Sentiment F**
50	**0.001**	0.0001	0.5472	0.3491
50	**0.0001**	0.0001	0.5774	0.3834
50	**0.00001**	0.0001	0.533	0.3374
50	**0.000001**	0.0001	0.533	0.3374
50	**0**	0.0001	0.533	0.3374

Fix Iteration, Word Learning Rate				
Iterations	**Word_lr**	**Trigram_lr**	**Entity F**	**Sentiment F**
50	0.0001	**0.1**	0.404	0.2119
50	0.0001	**0.01**	0.4332	0.2741
50	0.0001	0.001	0.5597	0.3557
50	0.0001	**0.0001**	0.5774	0.3834
50	0.0001	**0.00001**	0.5855	0.3794
50	0.0001	**0.000001**	0.5835	0.3812

Fix Trigram, Word Learning Rate				
Iterations	**Word_lr**	**Trigram_lr**	**Entity F**	**Sentiment F**
400	0.0001	0.00001	0.5806	0.3825
200	0.0001	0.00001	0.5885	0.377
100	0.0001	0.00001	0.5847	0.3805
50	0.0001	0.00001	0.5855	0.3794
40	0.0001	0.00001	0.5855	0.3794

Best Parameters for Perceptron				
Iterations	**Word_lr**	**Trigram_lr**	**Entity F**	**Sentiment F**
200	**0.0001**	**0.00001**	0.5885	0.377

FR Lower Case UNK2 – Viterbi only (no perceptron)

Iterations	Word_lr	Trigram_lr	Entity F	Sentiment F
1	0	0	0.5985	0.409

Fix Iteration, Trigram Learning Rate

Iterations	Word_lr	Trigram_lr	Entity F	Sentiment F
50	**0.001**	0.0001	0.6147	0.3452
50	**0.0001**	0.0001	0.6128	0.4133
50	**0.00001**	0.0001	0.6176	0.4181
50	**0.000001**	0.0001	0.6176	0.4181
50	**0**	0.0001	0.6176	0.4181

Fix Iteration, Word Learning Rate

Iterations	Word_lr	Trigram_lr	Entity F	Sentiment F
50	0.000001	**0.1**	0.3778	0.2267
50	0.000001	**0.01**	0.621	0.3983
50	0.000001	**0.001**	0.6223	0.4181
50	0.000001	**0.0001**	0.6176	0.4181
50	0.000001	**0.00001**	0.604	0.4158
50	0.000001	**0.000001**	0.5985	0.409

Fix Trigram, Word Learning Rate

Iterations	Word_lr	Trigram_lr	Entity F	Sentiment F
400	0.000001	0.001	0.5429	0.3527
200	0.000001	0.001	0.5742	0.378
100	0.000001	0.001	0.6205	0.42
50	0.000001	0.001	0.6223	0.4181
25	0.000001	0.001	0.6176	0.4181

Best Parameters for Perceptron

Iterations	Word_lr	Trigram_lr	Entity F	Sentiment F
50	**0.000001**	**0.001**	0.6223	0.4181

Part 5 Results (dev.p5.out)

EN
#Entity in gold data: 226
#Entity in prediction: 175

#Correct Entity : 108
Entity precision: 0.6171
Entity recall: 0.4779
Entity F: 0.5387

#Correct Sentiment : 69
Sentiment precision: 0.3943
Sentiment recall: 0.3053
Sentiment F: 0.3441

FR
#Entity in gold data: 223
#Entity in prediction: 198

#Correct Entity : 131
Entity precision: 0.6616
Entity recall: 0.5874
Entity F: 0.6223

#Correct Sentiment : 88
Sentiment precision: 0.4444
Sentiment recall: 0.3946
Sentiment F: 0.4181