Junyang Xin

N11176767

In my solution to this assingment, I have included 8 feature sets. They are listed below:

Feature set 1:

current=currentToken pos=currentTokenPOS

Feature set 2:

current=currentToken pos=currentTokenPos isInitialCap=True/False

Feature set 3:

prev=previousToken prevPos=previousTokenPOS current=currentToken pos=currentTokenPos isInitialCap=True/False next=nextToken nextPos=nextTokenPOS

Feature set 4:

prev=previousToken prevPos=previousTokenPOS current=currentToken pos=currentTokenPos isInitialCap=True/False isSuffixIng=True/False next=nextToken nextPos=nextTokenPOS

Feature set 5:

prev=previousToken prevPos=previousTokenPOS current=currentToken pos=currentTokenPos isInitialCap=True/False isSuffixIng=True/False isAllUpper=True/False isAllLower=True/False isMixedCase=True/False capInitWithPeriod=True/False endWithDigit=True/False hasHyphen=True/False next=nextToken nextPos=nextTokenPOS

Feature set 6:

prev=previousToken prevPos=previousTokenPOS current=currentToken pos=currentTokenPos isInitialCap=True/False isSuffixIng=True/False isAllUpper=True/False isAllLower=True/False isMixedCase=True/False capInitWithPeriod=True/False endWithDigit=True/False hasHyphen=True/False stemmed=stemmedCurrentToken next=nextToken nextPos=nextTokenPOS

Feature set 7:

prev=previousToken prevPos=previousTokenPOS prevTag=previousTokenTag current=currentToken pos=currentTokenPos isInitialCap=True/False isSuffixIng=True/False isAllUpper=True/False isAllLower=True/False isMixedCase=True/False capInitWithPeriod=True/False endWithDigit=True/False hasHyphen=True/False stemmed=stemmedCurrentToken next=nextToken nextPos=nextTokenPOS nextTag=nextTokenTag

Feature set 8:

prev=previousToken prevPos=previousTokenPOS prevTag=previousTokenTag current=currentToken pos=currentTokenPos isInitialCap=True/False isSuffixIng=True/False next=nextToken nextPos=nextTokenPOS nextTag=nextTokenTag

Besides the MaxEnt tagging, I also have implemented 2 versions of Viterbi and applied them with model trained with feature set 1 and model trained with features set 8, aka. model1 and model8 respectively. The simple version means that the posterior probablity is only calculated using the give feature set, and the complex version means that the posterior probability is calculated with the given feature set augmented with each state (i.e. I, B or O). The complex one should strictly mimic the MEMM in the textbook, and the simple one has some relaxation. However, the simple one shows identical performance of MaxEnt, but the complex one has way too worse performance. Please see results below.

Below are the results I got for each feature set. The measures I used are **number of correct tags** and **precision, recall, F0.5, F1, F2** of ngroup chunking.

Model1: Correct tags: 9186

Model2: Correct tags: 9186

Model3: Correct tags: 9356

Model4: Correct tags: 9355

Model5: Correct tags: 9355

Model6: Correct tags: 9351

Model7: Correct tags: 9358

Model8: Correct tags: 9363

Model1 with Viterbi-simple: Correct tags: 9186

Model1 with Viterbi-complex: Correct tags: 4480

Model8 with Viterbi-simple: Correct tags: 9363

Model8 with Viterbi-complex: Correct tags: 4497

Measures of model1: precision:0.8182861514919664 recall:0.8404715127701375 F1:0.8292304710215158 F0.5:0.8226290285362664 F2:0.8359387212755979

Measures of model2: precision:0.8182861514919664 recall:0.8404715127701375 F1:0.8292304710215158 F0.5:0.8226290285362664 F2:0.8359387212755979

Measures of model3: precision:0.8953307392996109 recall:0.9041257367387033 F1:0.8997067448680351 F0.5:0.8970760233918128 F2:0.9023529411764706

Measures of model4: precision:0.8954758190327613 recall:0.9021611001964637 F1:0.898806028577021 F0.5:0.896804937114288 F2:0.9008160703075958

Measures of model5: precision:0.8946957878315133 recall:0.9013752455795678

F1:0.898023096496379 F0.5:0.8960237481446761 F2:0.9000313873195229

Measures of model6: precision:0.8924814959096221 recall:0.900196463654224 F1:0.8963223787167449 F0.5:0.8940138921407943 F2:0.8986428179179414

Measures of model7: precision:0.8966731898238748 recall:0.900196463654224 F1:0.8984313725490197 F0.5:0.8973756365060712 F2:0.8994895956026698

Measures of model8: precision:0.8997247345654739 recall:0.8990176817288802 F1:0.89937106918239 F0.5:0.8995832350397105 F2:0.8991590033797061

Measures of model1 with Viterbi-simple: precision:0.8182861514919664 recall:0.8404715127701375 F1:0.8292304710215158 F0.5:0.8226290285362664 F2:0.8359387212755979

Measures of model1 with Viterbi-complex: precision:0.004557538928978352 recall:0.004715127701375246 F1:0.004634994206257241 F0.5:0.004588208304657031 F2:0.004682744088035589

Measures of model8 with Viterbi-simple: precision:0.8997247345654739 recall:0.8990176817288802 F1:0.89937106918239 F0.5:0.8995832350397105 F2:0.8991590033797061

Measures of model8 with Viterbi-complex: precision:0.005050505050505051 recall:0.005108055009823182 F1:0.005079117015042001 F0.5:0.0050619110661163464 F2:0.0050964403324447235

**Conclusion**: Feature conjunction does help improved the performance. Including prior and following states (I assume states here mean BIO tags) as features also helped improve the performance. The best performance I could get is with feature set 8 using MaxEnt and Viterbi-simple using feature set 8. And the strictly mimic Viterbi (i.e.Viterbi-complex) gives surprising low performance. Not sure if I have done anything wrong or not, but I really think I have done it the right way.