|  |  |
| --- | --- |
| **EX.NO: 12 NAMED ENTITY RECOGNITION USING GMB CORPUS** | |
| IN[1] | **import** pandas **as** pd  **import** numpy **as** np  **import** matplotlib.pyplot **as** plt  **import** seaborn **as** sns  sns**.**set(color\_codes**=True**)  sns**.**set(font\_scale**=**1)  **%matplotlib** inline  **%config** InlineBackend.figure\_format = 'svg'  **from** sklearn.cross\_validation **import** cross\_val\_predict, cross\_val\_score  **from** sklearn.ensemble **import** RandomForestClassifier  **from** sklearn\_crfsuite **import** CRF, scorers, metrics  **from** sklearn\_crfsuite.metrics **import** flat\_classification\_report  **from** sklearn.metrics **import** classification\_report, make\_scorer  **from** sklearn.grid\_search **import** RandomizedSearchCV  **import** scipy.stats  **import** eli5 |
| IN[2] | data **=** pd**.**read\_csv("../input/GMB\_dataset.txt", sep**=**"\t", header**=None**, encoding**=**"latin1") |
| IN[3] | data**.**head() |
| IN[4] | data**.**columns **=** data**.**iloc[0]  data **=** data[1:]  data**.**columns **=** ['Index','Sentence #','Word','POS','Tag']  data **=** data**.**reset\_index(drop**=True**)  data**.**head() |
| IN[5] | data**.**shape |
| IN[6] | data**.**info() |
| IN[7] | data**.**head() |
| IN[8] | **class** getsentence(object):    **def** \_\_init\_\_(self, data):  self**.**n\_sent **=** 1.0  self**.**data **=** data  self**.**empty **=** **False**  agg\_func **=** **lambda** s: [(w, p, t) **for** w, p, t **in** zip(s["Word"]**.**values**.**tolist(),  s["POS"]**.**values**.**tolist(),  s["Tag"]**.**values**.**tolist())]  self**.**grouped **=** self**.**data**.**groupby("Sentence #")**.**apply(agg\_func)  self**.**sentences **=** [s **for** s **in** self**.**grouped] |
| IN[9] | getter **=** getsentence(data)  sentences **=** getter**.**sentences  print(sentences[0]) |
| IN[10] | words **=** list(set(data["Word"]**.**values))  n\_words **=** len(words)  print(n\_words) |
| IN[11] | plt**.**style**.**use("ggplot")  plt**.**hist([len(s) **for** s **in** sentences], bins**=**50)  plt**.**show() |
| IN[12] | maxlen **=** max([len(s) **for** s **in** sentences])  print ('Maximum sentence length:', maxlen) |
| IN[13] | data**.**loc[data['Tag'] **==** 'B-org', 'Word']**.**head() |
| IN[14] | data**.**loc[data['Tag'] **==** 'I-org', 'Word']**.**head() |
| IN[15] | data**.**loc[data['Tag'] **==** 'B-per', 'Word']**.**head() |
| IN[16] | data**.**loc[data['Tag'] **==** 'I-per', 'Word']**.**head() |
| IN[17] | data**.**loc[data['Tag'] **==** 'B-geo', 'Word']**.**head() |
| IN[18] | data**.**loc[data['Tag'] **==** 'I-geo', 'Word']**.**head() |
| IN[19] | plt**.**figure(figsize**=**(15, 5))  ax **=** sns**.**countplot('Tag', data**=**data)  ax**.**set\_xticklabels(ax**.**get\_xticklabels(), rotation**=**90, ha**=**"center")  plt**.**tight\_layout()  plt**.**show() |
| IN[20] | plt**.**figure(figsize**=**(15, 5))  ax **=** sns**.**countplot('Tag', data**=**data**.**loc[data['Tag'] **!=** 'O'])  ax**.**set\_xticklabels(ax**.**get\_xticklabels(), rotation**=**90, ha**=**"center")  plt**.**tight\_layout()  plt**.**show() |
| IN[21] | plt**.**figure(figsize**=**(15, 5))  ax **=** sns**.**countplot('POS', data**=**data, orient**=**'h')  ax**.**set\_xticklabels(ax**.**get\_xticklabels(), rotation**=**90, ha**=**"center")  plt**.**tight\_layout()  plt**.**show() |
| IN[22] | **def** feature\_map(word):  **return** np**.**array([word**.**istitle(), word**.**islower(), word**.**isupper(), len(word),  word**.**isdigit(), word**.**isalpha()]) |
| IN[23] | words **=** [feature\_map(w) **for** w **in** data["Word"]**.**values**.**tolist()]  tags **=** data["Tag"]**.**values**.**tolist() |
| IN[24] | print(words[:5]) |
| IN[25] | pred **=** cross\_val\_predict(RandomForestClassifier(n\_estimators**=**20),X**=**words, y**=**tags, cv**=**5) |
| IN[26] | **from** sklearn.metrics **import** classification\_report  report **=** classification\_report(y\_pred**=**pred, y\_true**=**tags)  print(report) |
| IN[27] | **def** word2features(sent, i):  word **=** sent[i][0]  postag **=** sent[i][1]  features **=** {  'bias': 1.0,  'word.lower()': word**.**lower(),  'word[-3:]': word[**-**3:],  'word[-2:]': word[**-**2:],  'word.isupper()': word**.**isupper(),  'word.istitle()': word**.**istitle(),  'word.isdigit()': word**.**isdigit(),  'postag': postag,  'postag[:2]': postag[:2],  }  **if** i **>** 0:  word1 **=** sent[i**-**1][0]  postag1 **=** sent[i**-**1][1]  features**.**update({  '-1:word.lower()': word1**.**lower(),  '-1:word.istitle()': word1**.**istitle(),  '-1:word.isupper()': word1**.**isupper(),  '-1:postag': postag1,  '-1:postag[:2]': postag1[:2],  })  **else**:  features['BOS'] **=** **True**  **if** i **<** len(sent)**-**1:  word1 **=** sent[i**+**1][0]  postag1 **=** sent[i**+**1][1]  features**.**update({  '+1:word.lower()': word1**.**lower(),  '+1:word.istitle()': word1**.**istitle(),  '+1:word.isupper()': word1**.**isupper(),  '+1:postag': postag1,  '+1:postag[:2]': postag1[:2],  })  **else**:  features['EOS'] **=** **True**  **return** features |
| IN[28] | **def** sent2features(sent):  **return** [word2features(sent, i) **for** i **in** range(len(sent))]  **def** sent2labels(sent):  **return** [label **for** token, postag, label **in** sent] |
| IN[29] | X **=** [sent2features(s) **for** s **in** sentences]  y **=** [sent2labels(s) **for** s **in** sentences] |
| IN[30] | crf **=** CRF(algorithm**=**'lbfgs',  c1**=**0.1,  c2**=**0.1,  max\_iterations**=**100,  all\_possible\_transitions**=False**) |
| IN[31] | pred **=** cross\_val\_predict(estimator**=**crf, X**=**X, y**=**y, cv**=**5) |
| IN[32] | report **=** flat\_classification\_report(y\_pred**=**pred, y\_true**=**y)  print(report) |
| IN[33] | crf2 **=** CRF(algorithm**=**'lbfgs',  c1**=**10,  c2**=**0.1,  max\_iterations**=**100,  all\_possible\_transitions**=False**) |
| IN[34] | pred **=** cross\_val\_predict(estimator**=**crf2, X**=**X, y**=**y, cv**=**5)  report **=** flat\_classification\_report(y\_pred**=**pred, y\_true**=**y)  print(report) |
| IN[35] | crf2**.**fit(X, y) |
| IN[36] | labels **=** list(crf2**.**classes\_)  labels **=** list(filter(**lambda** a: a **!=** 'O', labels))  print(labels) |
| IN[37] | **%%time**  crf3 **=** CRF(  algorithm**=**'lbfgs',  max\_iterations**=**100,  all\_possible\_transitions**=True**  )  params\_space **=** {  'c1': scipy**.**stats**.**expon(scale**=**0.5),  'c2': scipy**.**stats**.**expon(scale**=**0.05),  }  f1\_scorer **=** make\_scorer(metrics**.**flat\_f1\_score,  average**=**'weighted', labels**=**labels)  rs **=** RandomizedSearchCV(crf, params\_space,  cv**=**3,  verbose**=**1,  n\_jobs**=-**1,  n\_iter**=**50,  scoring**=**f1\_scorer)  rs**.**fit(X, y) |
| IN[38] | print('Best parameters:', rs**.**best\_params\_)  print('Best CV score:', rs**.**best\_score\_)  print('Model size: {:0.2f}M'**.**format(rs**.**best\_estimator\_**.**size\_ **/** 1000000)) |
| IN[26] | sorted\_labels **=** sorted(  labels,  key**=lambda** name: (name[1:], name[0])) |
| IN[26] | crf3 **=** rs**.**best\_estimator\_  y\_pred **=** crf3**.**predict(X)  print(metrics**.**flat\_classification\_report(  y, y\_pred, labels**=**sorted\_labels, digits**=**3)) |
| IN[26] | crf3**.**fit(X,y) |
| IN[26] | eli5**.**show\_weights(crf3, top**=**30) |