NLP 課題: POS-tagging

廣瀬　惟歩

1　はじめに

自然言語処理タスクの一つに，Part-of-speech tagging (品詞タグ付け) というものが存在する．これは，文章中の各単語に品詞ラベルを割り当てるタスクであり，文法の解析に用いられる．

本課題では，特徴量を組み込んだ隠れマルコフモデル (HMM) を利用して各単語の品詞ラベルを学習，推定し，実験を通して精度を確認した．

2　手法

本課題で用いる手法は Graham Neubig 氏の NLP チュートリアル[*1]を基にした．本課題では以下の関数を要するプログラムを構築した:

■def train_POS(train_file):　品詞ラベルの遷移 (transition) と品詞ラベルの種類 (possible_tags)，各単語の品詞ラベル (emit) を記録するための関数．このうち transition と possible_tags は後の def HMM_viterbi で活用する．

■def train_feature(train_file):　train データから得た特徴量を基に重みを学習，更新するための関数．特徴量の計算には後の def create_feature を活用する．

■def HMM_viterbi(w, word):　ビタビアルゴリズム．def create_trans と def create_emit を活用して transition と emit の特徴量を計算し，それを基にスコアを計算することで，最終的に最短パスを導く．

■def create_feature(X, Y):　特徴量を計算し，値を返すための関数．def create_trans と def create_emit によって transition と emit の特徴量を計算する．

■def create_trans(prev_tag, next_tag):　transition の特徴量を計算するための関数．

■def create_emit(next_tag, word):　emit の特徴量を計算するための関数．

■def test_POS(test_file):　学習によって得られた重みパラメータを用いて，test データに対して品詞ラベルの予測を行うための関数．

■def accuracy(pred_list, answ_list):　テスト結果の精度 (accuracy) を計算するための関数．

[*1] https://github.com/neubig/nlptutorial/blob/master/download/12-struct/nlp-programming-ja-12-struct.pdf

3 実験

3.1 実験データ

train 用データには wiki-en-train.norm_pos[*2]を，test 用データには wiki-en-test.norm_pos[*3]を使用した．これらのデータでは各文章に対し，各単語とそれに対応する品詞ラベルが記載されている (例: In_IN, computational_JJ)．そのため，def train_POS と def test_POS では単語と品詞ラベルを_で分割し，重みパラメータと単語で品詞ラベルを学習，推測するようにした．

3.2 実験方法

train データと test データを用意したのち，コマンドラインから python feature-tagging.py を実行して学習とテストを行った．精度の確認方法は以下の通りである．

1. def test_POS において各文章を単語と品詞ラベルに分割し，品詞ラベルの方を Y_prime_test リストに追加する．
2. 単語と重みパラメータを def HMM_viterbi の入力とし，返された推測の品詞ラベルを Y_hat_test リストに追加する．
3. Y_prime_test リストと Y_hat_test リストを def accuracy の入力とし，2 つのリストの品詞ラベルの一致数を全体の品詞ラベル数で割ったものを精度とする．

4 結果

実験によって得られた精度は 0.522 であった．

[*2] https://github.com/neubig/nlptutorial/blob/master/data/wiki-en-train.norm_pos
[*3] https://github.com/neubig/nlptutorial/blob/master/data/wiki-en-test.norm_pos