KU KONKUK UNIVERSITY

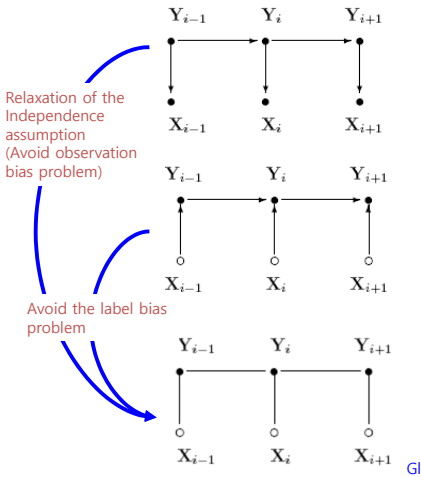
Statistical Models

HMM, MEMM, and CRFs

자연어처리연구실

PART-II

Graphical Models for Sequence Labeling



HMM: 결합확률 모델

$$p(x,y) = \frac{p(x \mid y)p(y)}{\sum_y p(x \mid y)p(y)}$$

1st order Markov Model

$$\hat{y} = \arg\max_{y \in S} \prod_{t=1}^{T} p(x_t \mid y_t)p(y_t \mid y_{t-1})$$

MEMM: 방향성 조건부 확률 모델

Feature function

$$p(y \mid x) = \frac{1}{Z(x)} \exp\left[\sum_{i=1}^{k} \lambda_i f_i(x,y)\right]$$

$$Z(x) = \sum_y \exp\left[\sum_{i=1}^{k} \lambda_i f_i(x,y)\right]$$

CRFs: 무방향성 조건부 확률 모델

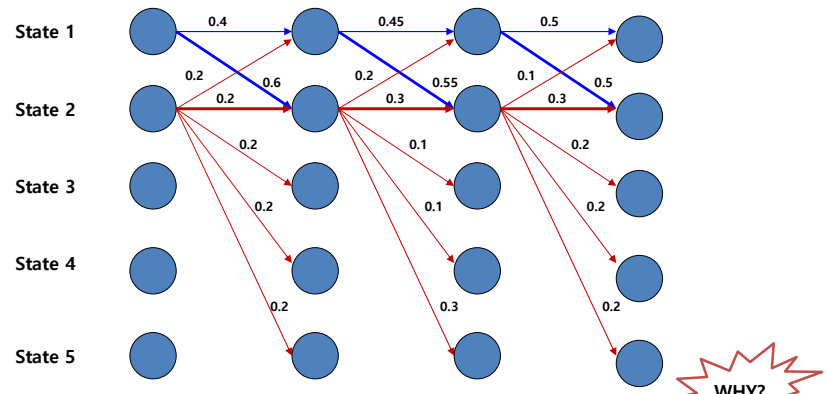
$$F_j(y,x) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, x, i)$$

Global feature function

$$p(y|x,\lambda) = \frac{1}{Z(x)} \exp\left(\sum_j \lambda_j F_j(y,x)\right)$$

Global normalization

Label Bias Problem of MEMM



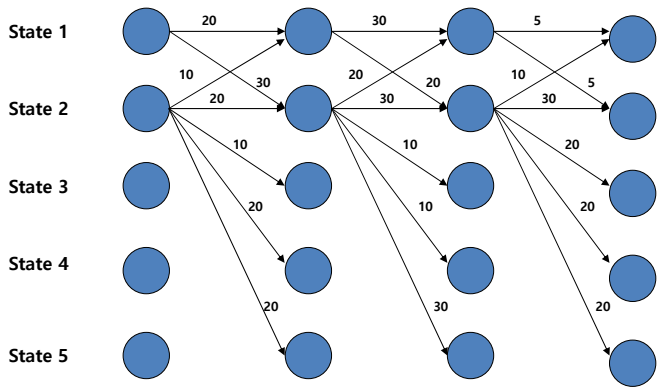
What the local transition probabilities say:

State 1 almost always prefers to go to state 2

State 2 almost always prefer to stay in state 2

1 → 2 → 1 → 2: 0.06
2 → 2 → 2 → 2: 0.018
1 → 1 → 1 → 1: 0.09

Global Normalization of CRFs

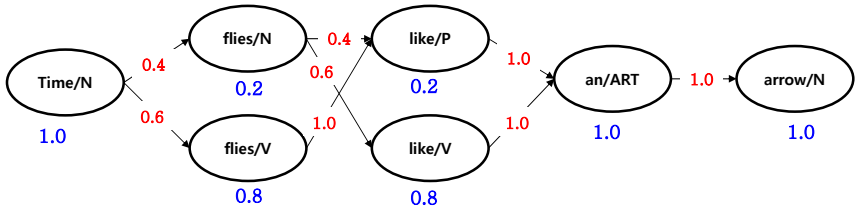


Local normalization: 전이가 많은 상태는 항상 불리

→ Global normalization (모든 상태를 기준으로 정규화)

확인 예제

- 다음 HMM을 바탕으로 "Time files like an arrow"라는 문장의 품사를 결정하는 최적의 경로를 비터비 알고리즘을 이용하여 구하시오.



실습

실습 코드 다운로드: https://github.com/KUNLP/Lecture

- CRFs를 이용하여 자동 띄어쓰기를 수행하는 프로그램을 작성하시오.
  - 입력(관측)
    - 한글 음절
  - 출력(레이블): B, I
    - B: 관측된 음절 앞에 공백을 추가해야 함을 나타내는 레이블
    - I: 관측된 음절 앞에 공백을 추가하지 말아야 함을 나타내는 레이블

출력: B I B I I B I

입력: 나 는 사 과 가 좋 아

  - 데이터 형식
    - 한글 음절 열 ₩t 레이블 열
    - 예제: 나 는 사 과 가 좋 아 ₩t B I B I I B I

실습

구글 colab 연결

```
from google.colab import drive
drive.mount("/gdrive", force_remount=True)
```

CRFs 라이브러리 설치

```
!pip install sklearn-crfsuite
```

```
Collecting sklearn-crfsuite
  Downloading https://files.pythonhosted.org/packages/25/74/5b7befa513482e6dee1f3dd68171a6c
Requirement already satisfied: tqdm>=2.0 in /usr/local/lib/python3.7/dist-packages (from sk
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sklearn-
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from skl
Collecting python-crfsuite>=0.8.3
  Downloading https://files.pythonhosted.org/packages/79/47/58f16c46506139f17de4630dbcfb877
     |████████████████████████████████| 747kB 5.4MB/s
Installing collected packages: python-crfsuite, sklearn-crfsuite
Successfully installed python-crfsuite-0.9.7 sklearn-crfsuite-0.3.6
```

실습

데이터 읽기 (학습 90%, 평가 10%)

```
import os
import sklearn_crfsuite
from sklearn_crfsuite import metrics


# 파일 경로
file_path = "/gdrive/My Drive/colab/crf/spacing_data.txt"

# "spacing_data.txt" 파일을 읽고 lines에 읽은 데이터를 저장
with open(file_path, "r", encoding="utf8") as inFile:
    lines = inFile.readlines()

# 데이터를 음절로 이루어진 문장과 정답 값으로 나누어 저장
datas = []
for line in lines:
    pieces = line.strip().split("₩t")
    eumjeol_sequence, label = pieces[0].split(), pieces[1].split()
    datas.append((eumjeol_sequence, label))

number_of_train_datas = int(len(datas)*0.9)

train_datas = datas[:number_of_train_datas]
test_datas = datas[number_of_train_datas:]

print("train_datas 개수 : " + str(len(train_datas)))
print("test_datas 개수 : " + str(len(test_datas)))
```

'₩t'으로 분리 후, 공백으로 분리하여 튜플을 구성한 후, 리스트를 구성

```
train_datas 개수 : 900
test_datas 개수 : 100
```


실습

데이터 변환 (자질 설계)

```
def sent2feature(eumjeol_sequence):
  features = []
  sequence_length = len(eumjeol_sequence)
  for index, eumjeol in enumerate(eumjeol_sequence):
      feature = { "BOS":False, "EOS":False, "WORD":eumjeol, "IS_DIGIT":eumjeol.isdigit() }

      if(index == 0):
          feature["BOS"] = True
      elif(index == sequence_length-1):
          feature["EOS"] = True

      if(index-1 >= 0):
          feature["-1_WORD"] = eumjeol_sequence[index-1]
      if(index-2 >= 0):
          feature["-2_WORD"] = eumjeol_sequence[index-2]

      if(index+1 <= sequence_length-1):
          feature["+1_WORD"] = eumjeol_sequence[index+1]
      if(index+2 <= sequence_length-1):
          feature["+2_WORD"] = eumjeol_sequence[index+2]

      features.append(feature)

  return features
```

나 는 사 과 가 좋 아

Feature	Value
BOS	False
EOS	False
WORD	사
IS_DIGIT	False
-1_WORD	는
-2_WORD	나
+1_WORD	과
+2_WORD	가

```
사 -> { "BOS":False, "EOS":False, "WORD":"사", "IS_DIGIT":False, "-2_WORD":"나", "-1_WORD":"는", "+1_WORD":"과", "+2_WORD":"가" }
```


실습

데이터 생성

```
train_x, train_y = [], []
for eumjeol_sequence, label in train_datas:
    train_x.append(sent2feature(eumjeol_sequence))
    train_y.append(label)

test_x, test_y = [], []
for eumjeol_sequence, label in test_datas:
    test_x.append(sent2feature(eumjeol_sequence))
    test_y.append(label)
```

CRFs 학습

```
crf = sklearn_crfsuite.CRF()
crf.fit(train_x, train_y)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:197: FutureWarning: From vers
  FutureWarning)
CRF(algorithm=None, all_possible_states=None, all_possible_transitions=None,
    averaging=None, c=None, c1=None, c2=None, calibration_candidates=None,
    calibration_eta=None, calibration_max_trials=None, calibration_rate=None,
    calibration_samples=None, delta=None, epsilon=None, error_sensitive=None,
    gamma=None, keep_tempfiles=None, linesearch=None, max_iterations=None,
    max_linesearch=None, min_freq=None, model_filename=None, num_memories=None,
    pa_type=None, period=None, trainer_cls=None, variance=None, verbose=False)
```


실습

CRFs 평가

Accuracy 계산

```
def show_predict_result(test_datas, predict):
  for index_1 in range(len(test_datas)):
      eumjeol_sequence, correct_labels = test_datas[index_1]
      predict_labels = predict[index_1]

      correct_sentence, predict_sentence = "", ""
      for index_2 in range(len(eumjeol_sequence)):
          if(index_2 == 0):
              correct_sentence += eumjeol_sequence[index_2]
              predict_sentence += eumjeol_sequence[index_2]
              continue

          if(correct_labels[index_2] == "B"):
              correct_sentence += " "
          correct_sentence += eumjeol_sequence[index_2]

          if (predict_labels[index_2] == "B"):
              predict_sentence += " "
          predict_sentence += eumjeol_sequence[index_2]

      print("정답 문장 : " + correct_sentence)
      print("출력 문장 : " + predict_sentence)
      print()
predict = crf.predict(test_x)
```

CRFs 실행

```
print("Accuracy score : " + str(metrics.flat_accuracy_score(test_y, predict)))
print()

print("10개의 데이터에 대한 모델 출력과 실제 정답 비교")
print()

show_predict_result(test_datas[:10], predict[:10])
```

```
Accuracy score : 0.8964135826020603

10개의 데이터에 대한 모델 출력과 실제 정답 비교

정답 문장 : 1914- 18년의 전쟁은 인류를 통합시킨 최초의 공통분모였다.
출력 문장 : 1914- 18년의 전쟁은 인류를 통합시킨 최초의 공통분 모였다.

정답 문장 : 하지만 이 전쟁은 죽음을 통해 인류를 통합시켰다.
출력 문장 : 하지만이 전쟁은 죽음을 통해 인류를 통합시켰다.
```

질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr

Edited by Harksoo Kim