Korean space correction

using

Conditional Random Field

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit/pycrfsuite_spacing

한국어 띄어쓰기 교정 문제

- 한국어 띄어쓰기 교정은 길이가 n 인 연속된 글자 $x_{1:n}$ 에 대하여 [띈다, 안띈다] 로 구성된 길이가 n 인 labels $y_{1:n}$ 을 부여하는 binary sequential labeling 문제입니다.

Sequential labeling

- Sequential labeling 은 $x = [x_1, x_2, \dots, x_n]$ 에 가장 적절한 $y = [y_1, y_2, \dots, y_n]$ 를 찾습니다.
  - 이를 확률모형으로 표현하면, $argmax_y\, P(y_{1:n}|x_{1:n})$ 입니다.

Potential function as Representation

- Conditional Random Field 는 Logistic regression 을 이용하는 sequential labeling 입니다.
- Categorical sequence 인 x 를 Logistic regression 이 이용하는 벡터로 표현하기 위하여 feature representation 변형합니다.
  - [이것, 은, 예문, 입니다] 와 같은 sequence 를 vector 로 표현합니다.
  - 이 역할을 하는 부분을 potential function 이라 합니다.

Potential function as Representation

- (x_{i-1}, x_i, y_{i-1}) 을 이용하는 품사 판별을 위하여 x_i 를 k 차원의 F_i 로 표현합니다.

$F_{i1} = \mathbf{1}$ if (x_{i-1} = '이것', x_i = '은', y_{i-1} = '명사') else **0**

$F_{i2} = \mathbf{1}$ if (x_{i-1} = '은', x_i = '예문', y_{i-1} = '조사') else **0**

…

$F_{ik} = \mathbf{1}$ if (x_{i-1} = '이', x_i = '단어', y_{i-1} = '조사') else **0**

Potential function as Representation

- Potential function 은 x_i 가 F_{ij} 와 같은지 Boolean 으로 표현하기 때문에 대부분의 값이 0 인 sparse vector 입니다.

F_{i2} = **1** if (x_{i-1} = '은', x_i = '예문', y_{i-1} = '조사') else **0**

Potential function as Representation

- 띄어쓰기 교정을 위하여 (x_{i-1}, x_i, y_{i-1}) 를 이용한다면,
  - "예문 입니다" 를 다음의 template을 이용
    - X[-1:0] : 앞글자와 현재글자
    - X[-1:0] & y[-1] : 앞글자와 현재글자, 앞글자의 띄어쓰기 정보
    - Y[-1] : 앞글자의 띄어쓰기 정보

단어 (feature)

빈도수

  - [[('x[0]=예', 1)],
    [('x[-1:0]=예문', 1), ('x[-1:0]=예문 & y[-1]=0', 1), ('y[-1]=0', 1)],
    [('x[-1:0]=문입', 1), ('x[-1:0]=문입 & y[-1]=1', 1), ('y[-1]=1', 1)],
    [('x[-1:0]=입니', 1), ('x[-1:0]=입니 & y[-1]=0', 1), ('y[-1]=0', 1)],
    [('x[-1:0]=니다', 1), ('x[-1:0]=니다 & y[-1]=0', 1), ('y[-1]=0', 1)]]

Potential function as Representation

- 마치 document – term frequency vector 처럼 해석할 수 있습니다.
  - [[('x[0]=예', 1)],
    [('x[-1:0]=예문', 1), ('x[-1:0]=예문 & y[-1]=0', 1), ('y[-1]=0', 1)],
    [('x[-1:0]=문입', 1), ('x[-1:0]=문입 & y[-1]=1', 1), ('y[-1]=1', 1)],
    [('x[-1:0]=입니', 1), ('x[-1:0]=입니 & y[-1]=0', 1), ('y[-1]=0', 1)],
    [('x[-1:0]=니다', 1), ('x[-1:0]=니다 & y[-1]=0', 1), ('y[-1]=0', 1)]]

char	Y	x[-1:0]=예문	x[-1:0]=예문 & y[-1]=0	'x[-1:0]=문입	x[-1:0]=문입 & y[-1]=1	..	y[-1] =0	y[-1] =1
예	0	0	0	0	0	..	0	0
문	1	1	1	0	0		1	0
입	0	0	0	1	1		0	1
니	0	0	0	0	0		1	0
다	1	0	0	0	0		1	0

Conditional Random Field

- Conditional Random Field 는 Softmax regression form 입니다.

Softmax regression	$P(y\|x) = \frac{exp(x^T \lambda_y)}{\sum_{y`} exp(x^T \lambda_{y`})}$
CRF	$P(y\|x) = \frac{exp(\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(x, i, y_i, y_{i-1}))}{\sum_{y`} exp(\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(x, i, y_i, y_{i-1}))}$

Conditional Random Field

- Conditional Random Field 의 자세한 설명은 다음 파일을 참고하세요.
  - topic_classification/from_logistic_to_crf.pdf

pycrfsuite

- crfsuite 는 c++ 로 개발된 CRF package 입니다. [1]
- pycrfsuite 는 crfsuite 를 Python 에서 이용할 수 있도록 도와줍니다.
  - Train, Tagging, Debugging 모듈의 options 은 crfsuite manual[2]
    을 보는 것이 좋습니다. 자세합니다.
  - Pycrfsuite official tutorial 에는 영어 데이터의 NER 예제가 있습니다. [3]

[1] http://www.chokkan.org/software/crfsuite/
[2] http://www.chokkan.org/software/crfsuite/manual.html
[3] https://github.com/scrapinghub/python-crfsuite/blob/master/examples/CoNLL%202002.ipynb

pycrfsuite

- 튜토리얼 작성 당시의 버전은 (0.9.5) 입니다. pip install 이 가능합니다.

```
pip install python-crfsuite
```

pycrfsuite

- pycrfsuite 의 default parameters 는 값들이 극단적입니다. 알고리즘을 적용할 때 잘 조절해야 합니다. 중요한 네 가지 패러매터를 소개합니다.

Parameter	Description
max_iterations	• 모델이 수렴할 때까지의 최대 반복횟수로, 작은 데이터에 대해서는 기본값 1,000 까지 필요하진 않습니다.
c1, c2	• 각각 L1, L2 regularization cost 로 기본값은 1 입니다. • L1 CRF 를 학습하고 싶다면 c2=0 으로 설정합니다.
feature.minfreq	• potential function 에 의하여 만들어진 feature 의 최소 빈도수입니다. Bag of words models 의 min count 와 같습니다. • 기본값이 0 이기 때문에, 적절하게 조절하지 않으면 지나치게 많은 features 를 이용합니다. • 이 문제 때문에 out of memory issue 가 자주 발생합니다.
verbose	• Verbose mode 를 제공합니다. • Verbose mode 에서 num of feature 등의 정보가 출력됩니다.

띄어쓰기 교정용 학습데이터

- 띄어쓰기 교정을 위하여, 띄어쓰기가 잘 지켜진 데이터의 패턴을 학습한 뒤, 띄어쓰기가 잘 되지 않은 문장에 적용합니다.

- 학습데이터를 구할 때 중요한 점은 vocabulary distribution 입니다.
  - 두번째가 띄어쓰기 노이즈 수준입니다.

띄어쓰기 교정용 학습데이터

- 띄어쓰기 교정을 하려는 데이터의 단어들이 자주 등장한 학습데이터가 필요합니다.
  - 채팅 데이터의 띄어쓰기 교정을 위해 뉴스 데이터는 유용하지 않습니다.
  - 채팅 데이터에서는 뉴스 데이터에서는 볼 수 없던 단어와 띄어쓰기 패턴들이 등장합니다.
  - 학습데이터와 교정할 데이터의 도메인이 맞아야 합니다.

띄어쓰기 교정용 학습데이터

- 오류란, 동일한 x 에 대하여 다수와 다른 y 입니다.
  - $P(y_1 \mid x) \ll P(y_2 \mid x)$ 라면 x 입장에서 y_1 은 노이즈입니다.
  - 학습데이터에서 "어서가요" (20%) vs. "어서 가요" (80%) 라면,
    "어서가요 → 어서 가요"로 교정합니다.

- 한국어 띄어쓰기 교정은, 오류가 적다고 판단되는 데이터를 선택한 뒤,
  자주 등장한 패턴으로 조금 등장한 패턴을 수정하는 문제입니다.

띄어쓰기 교정용 학습데이터

- 도메인이 같은 데이터는 우리가 분석할 데이터 자체입니다.
- 데이터에서 띄어쓰기가 어느 정도 지켜진 문장을 선택합니다.
  - 한국어의 어절의 평균 길이는 약 4음절입니다.
  - 띄어쓰기를 교정해야 하는 문서 집합에서 평균 어절 길이가 짧은 문장들을 선택하여 학습 데이터로 이용합니다.

pycrfsuite_spacing

CRF based Korean space corrector software

https://github.com/lovit/pycrfsuite_spacing

- pycrfsuite_spacing 은 pycrfsuite package 를 이용하여 미리 구현해둔 CRF 기반 한국어 띄어쓰기 교정기입니다.

  - 이 소프트웨어를 구현한 과정에서의 디테일은 다음 챕터 “Implementation using pycrfsuite” 에 있습니다.

  - 이 챕터에서는 pycrfsuite_spacing 의 사용법만 기술합니다.

- pycrfsuite_spacing 의 세 가지 모듈을 이용하여 띄어쓰기 교정기를 학습합니다.

  - pycrfsuite_spacing.TemplateGenerator
  - pycrfsuite_spacing.CharacterFeatureTransformer
  - pycrfsuite_spacing.sent_to_xy

pycrfsuite_spacing.TemplateGenerator

- TemplateGenerator 는 i 기준으로 이용할 앞/뒤 글자의 template 를 만듭니다.
  - 앞의 2 글자부터 현재 글자 / 앞, 뒤 한글자와 현재글자 / 현재부터 뒤의 2 글자를 이용하는 templates 이 만들어집니다.

```
from pycrfsuite_spacing import TemplateGenerator


TemplateGenerator(begin=-2, end=2, min_range_length=3, max_range_length=3).tolist()
```

```
[(-2, 0), (-1, 1), (0, 2)
```

pycrfsuite_spacing.CharacterFeatureTransformer

- CharacterFeatureTransformer 와 send_to_xy 는 templates 을 이용하여 각 글자를 x, y 로 변환합니다.

```
from pycrfsuite_spacing import CharacterFeatureTransformer
from pycrfsuite_spacing import sent_to_xy

templates = TemplateGenerator(begin=-2, end=2, min_range_length=3, max_range_length=3)
to_feature = CharacterFeatureTransformer(templates)

x, y = sent_to_xy('이것도 너프해 보시지', to_feature)
```

```
print(x)
```

```
[['X[0,2]=이것도'],
 ['X[-1,1]=이것도', 'X[0,2]=것도너'],
 ['X[-2,0]=이것도', 'X[-1,1]=것도너', 'X[0,2]=도너프'],
 ['X[-2,0]=것도너', 'X[-1,1]=도너프', 'X[0,2]=너프해'],
 ['X[-2,0]=도너프', 'X[-1,1]=너프해', 'X[0,2]=프해보'],
 ['X[-2,0]=너프해', 'X[-1,1]=프해보', 'X[0,2]=해보시'],
 ['X[-2,0]=프해보', 'X[-1,1]=해보시', 'X[0,2]=보시지'],
 ['X[-2,0]=해보시', 'X[-1,1]=보시지'],
 ['X[-2,0]=보시지']]
```

```
print(y)
```

```
['0', '0', '1', '0', '0', '1', '0', '0', '1']
```

Correct space after training

```
from pycrfsuite_spacing import PyCRFSuiteSpacing


model_path = 'my_model_path'
correct = PyCRFSuiteSpacing()
correct.train(docs, model_path)
correct.correct('이건진짜좋은영화라라랜드진짜좋은영화')
```

```
'이건 진짜 좋은 영화 라라랜드 진짜 좋은 영화'
```

Correct space with trained model

```
from pycrfsuite_spacing import PyCRFSuiteSpacing


model_path = 'my_model_path'
correct = PyCRFSuiteSpacing()
correct.load_tagger(model_path)
correct.correct('이건진짜좋은영화라라랜드진짜좋은영화')
```

```
'이건 진짜 좋은 영화 라라랜드 진짜 좋은 영화'
```

Implementation using pycrfsuite

Implementation using pycrfsuite

- pycrfsuite 를 이용하여 한국어 띄어쓰기 교정기를 만드는 과정입니다.
- pycrfsuite package 를 이용할 때 주의해야 하는 점들을 살펴봅니다.

pycrfsuite potential function

- sent = '이것도 너프해 보시지' 에 대하여 다음과 같은 x, y 를 만듭니다.

```
x = [['X[0,2]=이것도'],
     ['X[-1,1]=이것도', 'X[0,2]=것도너'],
     ['X[-2,0]=이것도', 'X[-1,1]=것도너', 'X[0,2]=도너프'],
     ['X[-2,0]=것도너', 'X[-1,1]=도너프', 'X[0,2]=너프해'],
     ['X[-2,0]=도너프', 'X[-1,1]=너프해', 'X[0,2]=프해보'],
     ['X[-2,0]=너프해', 'X[-1,1]=프해보', 'X[0,2]=해보시'],
     ['X[-2,0]=프해보', 'X[-1,1]=해보시', 'X[0,2]=보시지'],
     ['X[-2,0]=해보시', 'X[-1,1]=보시지'],
     ['X[-2,0]=보시지']]
y = ['0', '0', '1', '0', '0', '1', '0', '0', '1']
```

pycrfsuite potential function

- sent = '이것도 너프해 보시지' 를 x, y 로 만드는 함수를 구현해 둡니다.

```
x, y = sent_to_xy('이것도 너프해 보시지')
```

pycrfsuite Trainer

- pycrfsuite.Trainer 에 (x, y) pair 를 입력합니다.

```
import pycrfsuite


trainer = pycrfsuite.Trainer(verbose=true)
for sent in docs:
    x, y = sent_to_xy(sent)
    trainer.append(x,y)
```

pycrfsuite Trainer

- pycrfsuite.Trainer 의 parameters 를 조절합니다.
  - 그 외의 parameters 는 crfsuite manual 을 참고하세요.

```
import pycrfsuite


trainer = pycrfsuite.Trainer(verbose=true)
Trainer.set_params({'feature.minfreq': 3,
                    'max_iterations': 100 })
```


http://www.chokkan.org/software/crfsuite/manual.html

pycrfsuite Trainer

- train() 을 위해 model path 를 입력해야 합니다.

```
import pycrfsuite


trainer = pycrfsuite.Trainer(verbose=true)
for sent in docs:
    x, y = sent_to_xy(sent)
    trainer.append(x,y)

model_path = 'my_model_path'
trainer.train(model_path)
```

pycrfsuite Trainer

- Training 이전에 feature selection 을 해야 합니다.
  - trainer.set_params({"'feature.minfreq': 3}) 에 의하여 feature 의 숫자를 줄이는 것은 이미 메모리에 모든 features 가 올라간 뒤 입니다.
  - Feature filtering 을 하지 않으면 trainer.append(x, y) 중에 메모리가 지나치게 증가합니다.

```
import pycrfsuite


trainer = pycrfsuite.Trainer(verbose=true)
for sent in docs:
    x, y = sent_to_xy(sent)
    x = my_filter(x)
    trainer.append(x,y)
```

pycrfsuite Trainer

```
import pycrfsuite


trainer = pycrfsuite.Trainer(verbose=true)
For sent in docs:
    x, y = sent_to_xy(sent)
trainer.append(x,y)
```

pycrfsuite Tagger

- 학습된 모델을 이용하기 위해서 pycrfsuite.Tagger 를 이용합니다.

```
import pycrfsuite


model_path = 'my_model_path'
tagger = pycrfsuite.Tagger(model_path)
```

pycrfsuite Tagger

- 띄어쓰기 교정을 위하여 입력된 문장을 x 로 변환합니다.

```
import pycrfsuite


model_path = 'my_model_path'
tagger = pycrfsuite.Tagger(model_path)


x, _ = sent_to_xy(sent)
y_pred = tagger.tag(x)
```